HOCHSCHULE DER MEDIEN

IBM

Diplomarbeit

im Studiengang Medieninformatik

z/VSE

Design und Implementierung einer Java Klassenbibliothek zur Abbildung von security-relevanten Parametern im z/VSE

vorgelegt im
Sommersemester 2007
von
Matthias Herbert
Matrikelnummer: 13873
an der Hochschule der Medien Stuttgart

zur Erlangung des akademischen Grades
Diplom-Ingenieur (FH)
Fachrichtung Medieninformatik

Erstprüfer:	Prof. Walter Kriha Hochschule der Medien
Zweitprüfer:	Dipl.-Inf. Jörg Schmidbauer IBM Deutschland Entwicklung GmbH z/VSE Development
Eingereicht am:	30. August 2007

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Böblingen, den 30. August 2007

Matthias Herbert

Matrikelnummer 13873

Abstract

Die vorliegende Diplomarbeit wurde im IBM Entwicklungslabor in Böblingen für die Abteilung z/VSE Development geschrieben. Ausgangspunkt ist das Fehlen einer Möglichkeit der zentralisierten und automatisierten Überwachung der Systemsicherheit von VSE. Diese Arbeit soll eine Lösung dieses Problems in Form einer Java Klassenbibliothek bereitstellen, mit deren Hilfe alle sicherheits-relevanten Parameter und Einstellungen des Systems ausgelesen werden können. Diese Daten sollen von der Java Klassenbibliothek zusammengeführt, abgeglichen und dem Programmierer strukturiert, im Sinne der Objektorientierung, zur Verfügung gestellt werden. Diesee Lösung wird z/VSE Kunden in Zukunft die Möglichkeit bieten, zu jeder Zeit die komplexen Sicherheitseinstellungen ihre Systeme automatisiert zu überwachen.

Danksagungen

Hiermit möchte ich allen danken, die mich während der Erstellung meiner Diplomarbeit bei der IBM Deutschland Entwicklung GmbH in Böblingen unterstützt und begleitet haben.

Als erstes möchte ich mich beim z/VSE Development Team bedanken, das mir immer hilfsbereit mit Rat und Tat zur Seite stand. Besonderen Dank möchte ich meinem Betreuer Herrn Jörg Schmidbauer für seine Unterstützung, Hilfsbereitschaft, Durchsichten meiner Arbeit und seiner persönlichen Betreuung aussprechen. Weiter möchte ich Herrn Karsten Kraul für seine Hilfe und den informativen Erfahrungsaustausch danken.

Sehr dankbar bin ich Herrn Walter Kriha für die jahrelange Unterstützung und Hilfsbereitschaft während meines Studiums und seiner Betreuung während meiner Diplomarbeit von Seiten der Hochschule.

Abschließend möchte ich mich bei meinen Eltern und Freunden bedanken, die mir während dieser Zeit immer eine große Hilfe und Motivation waren.

Inhaltsverzeichnis

1 Einleitung

Diese Diplomarbeit wurde in der Abteilung 'z/VSE Development' der IBM Deutschland Entwicklung GmbH in Böblingen geschrieben. Für die automatische Überwachung der Sicherheitseinstellungen des Betriebssystems z/VSE, das von dieser Abteilung entwickelt wird, existierte bisher keine Lösung. Während dieser Arbeit wird eine Java Klassenbibliothek entwickelt die alle sicherheits-relevanten Parameter aus dem VSE ausliest und die Daten über eine Java Schnittstelle zur Verfügung stellt.

Diese sicherheits-relevanten Informationen können dann beispielsweise über das Produkt Tivoli Security Compliance Manager (TSCM) zum Zwecke eines State Monitorings regelmäßig und automatisiert auf Übereinstimmug mit firmen-spezifischen Vorgaben geprüft werden. Des Weiteren soll die Klassenbibliothek schon intern verschiedene Daten zusammenführen und vergleichen, so dass der Zustand des Betriebssystems bezüglich der Sicherheit schnell und effizient kontrolliert werden kann.

1.1 Ziele dieser Diplomarbeit

Die Ziele dieser Arbeit wurden durch einen VSE Kunden vorgegeben und im Laufe der Entwicklung erweitert und ausgebaut. Die wesentlichen Vorgaben sind hierbei:

- Zugriff auf das z/VSE Betriebssystem über eine Java Schnittstelle
- Zugriff auf alle Sicherheitsparameter und Einstellungen
- Aufbereitung der Daten, d.h. Aggregierung von Daten aus unterschiedlichen Quellen im VSE
- Erstellung einer Java Klassenbibliothek

Neben diesen funktionalen Zielen müssen weitere technische und strukturelle Ziele erreicht werden, die nach dem *Rational Unified Process* (beschrieben in [ZGK04]) in klar strukturierte Abschnitte des Softwareentwicklungsprozesses unterteilt sind. Nachdem die Anforderungen in der Analysephase festgelegt werden, folgen die Designentscheidungen, die sich an Prinzipien der Objektorientierung halten.

1.2 Dokumentenstruktur

Im Folgenden wird eine kurze Übersicht über die Struktur der Arbeit und den Inhalt der einzelnen Kapitel gegeben.

Kapitel Motivation

In diesem Kapitel wird die allgemeine Notwendigkeit von IT-Sicherheit im Unternehmen und die Überwachung von innerbetrieblichen und betriebssysteminternen sicherheitsrelevanten Parametern diskutiert.

Kapitel Grundlagen

Die zum Verständnis dieser Arbeit vorausgesetzten Themen werden dem Leser in diesem Kapitel kurz erläutert. Das Kapitel gibt eine Einführung in die Welt der Mainframes (Großrechner) und deren verschiedene Betriebssysteme, bis hin zu einigen in dieser Arbeit angesprochenen Tivoli Produkten.

Kapitel IBM Betriebssystem z/VSE

Dieses Kapitel vertieft VSE spezifische Grundlagen und gibt einen Einblick in die geschichtliche Entwicklung von VSE und liefert eine detaillierte Darstellung der VSE Systemkomponenten. Weiter wird auf die Anbindung von VSE an die Außenwelt und die Datenspeicherung im VSE eingegangen.

Das VSE Sicherheitskonzept

Das komplexe Sicherheitskonzept von VSE, welches eines der Grundpfeiler dieser Arbeit darstellt, wird in diesem Kapitel detailliert beschrieben und erklärt.

Kapitel Abgrenzung

Hier wird darauf eingegangen, wie sich die vorliegende Arbeit gegenüber benachbarten Themengebieten abgrenzt. Weiter werden verschiedene Sicherheitskonzepte von Betriebssystemen gegenübergestellt und kurz verglichen.

Kapitel Analyse

Dieses Kapitel beschreibt die Analysephase der Entwicklung und definiert alle Anforderungen an die zu entwickelnde Java Klassenbibliothek.

Kapitel Design

In diesem Kapitel werden die Designentscheidungen erklärt und detailliert ausgeführt. Des Weiteren wird die Klassenstruktur und die Bedeutung der wichtigsten Klassen der Bibliothek erläutert.

Kapitel Realisierung

Hier werden für die Implementierung einige Methoden und deren Funktionalität exemplarisch aufgezeigt.

Kapitel Zusammenfassung und Ausblick

Dieses Kapitel fasst die Leistung und den Mehrwert dieser Arbeit kurz zusammen und liefert einen Ausblick für die verschiedenen Verwendungsmöglichkeiten der entwickelten Java Klassenbibliothek VSESecurity.

2 Motivation

2.1 Bedeutung von Security in Unternehmen

Die Bedeutung von Security reicht von (Daten-) Sicherheit über Betriebsschutz bis hin zu Gefahrlosigkeit, wobei in dieser Arbeit in erster Linie die Sicherheit des Betriebssystems z/VSE behandelt wird.

Rein unternehmerisch betrachtet wird IT-Security als Mechanismus zum allgemeinen Schutz der Unternehmenswerte verstanden. Als die wichtigsten Unternehmenswerte sind hier die Kundendaten zu nennen. Hier definiert sich Security speziell durch den Schutz dieser sensiblen Daten vor nicht autorisierten Zugriffen, durch eine möglichst hohe Ausfallsicherheit der Systeme und durch *recovery*[1] Systeme, die im Falle eines Absturzes oder Angriffs den Datenverlust minimal halten sollen.

IT-Security muss außerdem gewährleisten, dass die zu schützenden Daten kontrolliert und eingeschränkt einem ausgewählten Kreis an autorisierten Personen oder Programmen zugänglich sind, da sonst kein sicheres und profitables Arbeiten mit den Informationen möglich ist. Dabei muss festgestellt werden, wer wie lange welche Art von Zugriff auf Daten haben darf und welche Informationen internen und externen Mitarbeitern bzw. Geschäftspartnern zur Verfügung stehen. Diese Entscheidung ist abhängig von dem Wert der zu schützenden Ressource, der sich grob aus den Kosten errechnet, die sich bei Verlust oder nicht Erreichbarkeit der Daten für das Unternehmen ergeben. Aus diesen Kosten und den Mitteln, die man zum Schutz der Information aufwenden müsste, erhält man eine Klassifizierung der Daten, die die Intensität der Schutzmaßnahmen definiert.

Informationen gelten generell erst dann als sicher geschützt und verwendbar wenn die Vertraulichkeit (Confidentiality), Vollständigkeit (Integrity) und Verfügbarkeit (Availability) sichergestellt ist.

- Confidentiality: Schutz der Daten vor nicht autorisierten Zugriffen oder Offenlegung

[1] Wiederherstellung von zerstörten oder verlorenen Daten

- Integrity: Daten sind fehlerfrei und wurden nicht verändert
- Availability: Daten sind verfügbar

Betrachtet man nun die Metaebene der sicheren Verwaltung von Unternehmensdaten, nämlich die Sicherstellung der Integrität des Datenverarbeitungs-Systems (Betriebssystem) selbst, kommt man zum Begriff des *State Monitoring.* Hierbei sollen security-relevante Systemeinstellungen jederzeit auf standardisierte Art und Weise abrufbar sein, um die Sicherheit des IT-Systems überprüfen und kontrollieren zu können. Die vorliegende Arbeit stellt einen auf Java basierenden Ansatz vor, der diese Anforderungen erfüllt.

3 Grundlagen

3.1 IBM Mainframes als Hardware Plattform

Als Mainframes werden üblicherweise Großrechner der IBM S/360, S/390 und spätere Baureihen bezeichnet. Heutige Vertreter dieser Maschinen sind die Modelle der *System z*[1] Generation. Die grundlegende Philosophie von Großrechnern basiert auf der Fähigkeit, sehr große Datenmengen und Transaktionen sicher und verlässlich zu verarbeiten. Nur durch eine gut durchdachte Architektur ist es möglich, solchen Anforderungen gerecht zu werden.

Hervorzuheben ist hier, dass auf Mainframes mehrere Prozessoren gleichzeitig laufen und deshalb eine, ohne durch I/O Befehle unterbrochene, parallele Verarbeitung von Anwendungsprogrammen möglich wird. Diese wird erreicht, indem ein einzelner Prozessor allein für die Ein- und Ausgabeverarbeitung (I/O) zuständig ist, so dass die I/O von der eigentlichen Datenverarbeitung getrennt ist.

Des Weiteren können logische Partitionen, so genannte LPARs (siehe Abbildung 4.1 auf Seite 27), aufgesetzt werden. Jeder LPAR verhält sich dann wie ein eigenständiger Mainframe mit einem eigenen Betriebssystem. Solchen LPARs kann manuell eine feste Anzahl an Prozessoren zugeteilt werden, die Verwaltung und Zuweisung von Prozessoren kann aber auch dem System überlassen werden. Diese Technik ermöglicht es, auf einem einzigen Mainframe mehrere verschiedene sowie unabhängige Betriebssysteme parallel zu betreiben. Die Anzahl der möglichen LPARs ist hierbei abhängig vom Großrechnermodell und von Faktoren wie Speichergröße und Prozessorleistung.

Mainframes bieten ein sehr hohes Maß an Ausfallsicherheit, Verfügbarkeit, Wartbarkeit, Skalierbarkeit, Sicherheit und Kompatibilität zu älteren Softwareversionen. Um dies gewährleisten zu können, sind Mainframes komplett redundant aufgebaut, so dass einzelne Software- beziehungsweise Hardwarefehler oder -ausfälle keinen Einfluss auf die Systemverfügbarkeit haben.

[1] weitere Informationen sind unter http://www.ibm.com/systems/z/ zu finden

Der Anwendungsbereich von Mainframes lässt sich in zwei Kategorien einteilen. Zum Einen werden Großrechner eingesetzt, um große Stapelverarbeitungen (Batch Jobs) abzuarbeiten, zum Anderen, um Online Prozesse durchzuführen.

Stapelverarbeitungen laufen im Hintergrund ab und benötigen keine weitere Aktion von Seiten des Benutzers. Mit ihnen werden riesige Datenmengen verarbeitet und Output, der dann zur Weiterverarbeitung bereit steht oder ausgedruckt wird, generiert.

Im Falle der Online Prozesse arbeitet eine große Anzahl von Benutzern gleichzeitig mit dem Betriebssystem und dem Mainframe. Hierbei wird auf unternehmenskritischen Anwendungen eine unbestimmte Menge an verschiedenen Transaktionen parallel ausgeführt. Online Transaktionen zeichnen sich dadurch aus, dass die Kommunikation zwischen Anwender und System sehr kurz ist und jeder Anwender sofort eine Reaktion sowie kurze Antwortzeiten erwartet.

Im Folgenden werden die verschiedenen Großrechner Betriebssysteme kurz beschrieben.

3.2 IBM Mainframe Betriebssysteme

Den Beginn im Bereich der Großrechnersysteme machte OS/360 im Jahre 1964. Anschließend durchlief das System, das als erstes Zugriff auf externe Massenspeicher voraussetzte, drei Stufen der Entwicklung. Die erste Version unterstützte nur eine rein sequenzielle Jobverarbeitung. In der nächsten Version konnte durch Pseudomultitasking[2] und eine manuelle Speicherzuweisung eine fest definierte Menge an Jobs gleichzeitig ausgeführt werden. Mit der Einführung einer 'Job Control Language' (JCL[3]), ähnlich einer *system interpreter* Sprache[4] neuerer Systeme, in der letzten Version, wurde es schließlich möglich, den Ablauf und die Ausführung einer Software im Voraus festzulegen. Auch konnten alle physikalischen Informationen des auszuführenden Jobs in JCL ausgelagert werden. Mit dieser Sprache lassen sich komplexe Abläufe (Jobs) granular und flexibel planen.

Obwohl die Geburt des Mainframes schon über 40 Jahre zurückliegt, erwachsen die aktuellen Großrechnerbetriebssysteme weiterhin aus den Wurzeln des OS/360. In der Zwischenzeit wurden mehrere verschiedene Betriebssysteme für Mainframes entwickelt und

[2] Prozesse oder Ereignisse werden nach einem fixen Zeitplan Prozessoren zugeteilt, die kein Multitasking unterstützen

[3] weitere Informationen zu JCL sind unter http://www.okstate.edu/cis_info/cis_manual/jcl_toc.html zu finden

[4] Software, die Quellcode zur Laufzeit analysiert und ausführt

haben sich in der Wirtschaft etabliert. Die Wichtigsten werden im Nachstehenden kurz vorgestellt und beschrieben.

3.2.1 z/OS

z/OS ist das mächtigste Betriebssystem für Mainframes. Im Vergleich zu anderen Mainframe Betriebssystemen ist es zwar sehr komplex, dafür aber entsprechend leistungsfähiger. z/OS ist darauf ausgelegt, Anwendungen eine sichere und vor allem hoch verfügbare Umgebung bereitzustellen. Eines der wichtigsten Unterscheidungsmerkmale des z/OS gegenüber Betriebssystemen auf Nicht-Mainframe Plattformen ist die Jobverarbeitung. Im Gegensatz zu Systemen wie Windows oder Linux behandelt z/OS Anfragen und Jobaufträge nicht als eine Einheit, die von Anfang bis Ende im Prozessor bleibt, sondern reagiert dynamisch auf Wartezeiten oder I/O Inputs. Sobald ein Prozess auf weitere Informationen warten muss, um fortzufahren, speichert z/OS alle notwendigen Daten ab und lässt während der Wartezeit einen anderen Prozess im Prozessor laufen. Zusätzlich wird jede Anfrage in kleinere Teile zerlegt, die wiederum von unabhängigen Komponenten (Prozessoren) bearbeitet werden, so dass Ergebnisse schneller berechnet werden können. Mit diesem System werden die Prozessoren und die komplette Rechenleistung optimal ausgenutzt und es entstehen keine überflüssigen Leerlaufzeiten.

Darüber hinaus bietet z/OS inzwischen eine Java und C++ Runtime und *Unix System Services* an. Diese Erweiterungen simulieren ein hierarchisches Filesystem (HFS) und machen es möglich UNIX Programme wie *SAP R3* oder *Web Application Server* unter z/OS zu betreiben.

3.2.2 z/VSE-Virtual Storage Extended

Der kleine Bruder des z/OS Betriebssystems, der speziell für den Mittelstandsbereich entwickelt wurde, ist optimiert für Batch Jobs und Transaktionen. Wie auch z/OS ist dieses Betriebssystem geeignet, mehrere Batch Jobs und umfassende Transaktionen parallel auszuführen. Transaktionen werden im z/VSE (ebenso im z/OS) vom *Customer Information Control System Transaction Server (CICS TS* siehe Abschnitt 4.2.2 auf Seite 30) ausgeführt. Im Gegensatz zum z/OS, welches verschiedene Runtimes und ein HFS anbietet, verfolgt VSE eine andere Strategie. Hier steht die Verbindungsfähigkeit (Connectivity) nach außen im Vordergrund. Das heißt auf VSE können keine UNIX Programme direkt ausgeführt werden, jedoch kann VSE mit einem z/Linux (siehe nächster Abschnitt 3.2.4),

das parallel auf demselben Mainframe (in einem anderen LPAR) installiert ist, sehr schnell über HyperSockets[5] kommunizieren. Mit dieser Kombination bietet VSE die Funktionalität des z/OS UNIX Programme auszuführen ohne dessen komplexen Administrationsaufwand.

Detailliertere Informationen zur Geschichte und der technischen Entwicklung von z/VSE sind im Kapitel 4.1 auf Seite 27 zu finden.

3.2.3 z/VM-Virtual Machine

VM war das erste Betriebssystem welches Virtualisierung erlaubte. Hierbei wird eine Unterteilung der vorhandenen Hardware-Ressourcen in eigenständige, voneinander unabhängige virtuelle Maschinen (Server) realisiert, die die Mainframe-Hardware oder nicht installierte Hardware emulieren. In virtuellen Maschinen (VMs) können dann die verschiedensten Betriebssysteme installiert werden, die alle parallel und isoliert voneinander auf dem Großrechner betrieben und gepflegt werden können. Der parallele Zugriff der virtuellen Server auf die Mainframe-Hardware wird vom z/VM kontrolliert. Mit Hilfe der Virtualisierungstechnik wird eine effizientere und kostengünstigere Nutzung der Ressourcen erreicht.
Im Zusammenspiel mit vielen zLinux (siehe nächsten Abschnitt) Installationen unter VM ergeben sich heutzutage sehr interessante Einsatzmöglichekiten im Bereich Web-Hosting. Hierbei können virtuelle Server (Linux Images) innerhalb von Sekunden bereitgestellt bzw. wieder entfernt werden.

Im z/VM können die folgenden Betriebssysteme installiert werden:

- z/VM, d.h. es ist eine Virtualisierung von VM unter VM möglich
- z/OS
- z/VSE
- Linux for System z
- z/TPF
- Conversational Monitor System (CMS), wobei CMS kein eigenständiges Betriebssystem ist und ausschließlich unter VM einsetzbar ist.

3.2.4 Linux on System z

In den letzten Jahren wurde bei verschiedenen Linux Distributionen der Kernel so umgeschrieben, dass Linux auf Mainframes betrieben werden kann. Es kann sowohl nativ

[5] HyperSockets sind integrierte Netzwerkkomponenten, die durch ein simuliertes Ethernet LAN die Kommunikation zwischen den LPARs eines Mainframes ermöglichen.

auf LPARs betrieben werden als auch unter z/VM nahezu beliebig oft installiert werden. Außerdem ist Linux on System z zu der 64-Bit Architektur des Großrechners kompatibel.

In den folgenden Abschnitten werden verschiedene Tivoli Produkte beschrieben, soweit sie für die vorliegende Arbeit relevant sind.

3.3 Tivoli

Unter dem Markennamen Tivoli werden mehrere Softwareprodukte der IBM zusammengefasst, die allgemein ausgedrückt, darauf ausgelegt sind, IT- und Informationssysteme zu verwalten, zu automatisieren und durch festgelegte Prozesse zu unterstützen.[6] Außerdem bietet Tivoli Funktionen zum *State Monitoring* (siehe Abschnitt 3.3.4 auf Seite 25), die in Verbindung mit der in dieser Arbeit entwickelten Java Klassenbibliothek zur Überwachung von VSE Sicherheitseinstellungen eingesetzt werden können.

Im Folgenden werden die Produkte Tivoli Security Compliance Manager, Tivoli Risk Manager und das Tivoli Sicherheitskonzept näher beschrieben. Abschließend werden die Unterschiede von *Event* und *State Monitoring* erläutert.

3.3.1 Tivoli Sicherheitskonzept

Das Tivoli Sicherheitskonzept setzt sich aus Zugriffskontrollen auf Ressourcen, Einhaltung unternehmensweiter Sicherheitsrichtlinien und Lokalisierung und Beseitigung von Sicherheitslücken in Unternehmens-Software zusammen. In den folgenden Unterkapiteln werden die einzelnen Tivoli Komponenten ausführlicher vorgestellt.

3.3.2 Tivoli Security Compliance Manager (TSCM)

Der TSCM wurde konzipiert, um die Einhaltung aller z.B. durch die Firma vorgeschriebenen Sicherheitsrichtlinien für Betriebssysteme (security policies) in einer IT-Landschaft zu gewährleisten und mögliche sicherheits-relevante Schwachstellen zu erkennen.

Der Anwender, in der Regel ein Systemadministrator, bemerkt die Schwere des Verstoßes oder des Schwachpunkts an der farblichen Kennzeichnung der jeweiligen Richtlinie

[6] weitere Informationen über die Produktfamilie Tivoli und deren Anwendungsszenarien sind unter http://www.ibm.com/software/tivoli/ zu finden.

und kann sofort die notwendigen Maßnahmen einleiten. Neben der zentralisierten manuellen Kontrolle durch einen Administrator fragt der Compliance Manager in regelmäßigen Abständen alle Sicherheitseinstellungen bei allen angeschlossenen (Betriebs-) Systemen automatisch ab.

Die Sicherheitsrichtlinien und deren Toleranzen sind durch ein vordefiniertes Template zu wählen oder frei konfigurierbar.

Die bis heute unterstützten Betriebssysteme beschränken sich auf:

- AIX
- HP-UX
- Linux
- SUN Solaris
- Windows 95 / 98 / 2000 / NT / XP

Im Rahmen dieser Arbeit wird eine Java Klassenbibliothek erstellt, über die der TSCM auf alle sicherheits-relevanten Parameter des Betriebssystems z/VSE zugreifen kann.

3.3.2.1 TSCM Collectors

Ein TSCM Collector ist ein in Java geschriebenes Software-Modul, das als Java-Archiv Datei (client.jar) in den TSCM eingebunden wird. Die durch den Collector bereit gestellten Klassen und Methoden liefern alle benötigten sicherheits-relevanten Informationen und Parameter eines mittels des TSCM zu prüfenden Betriebssystems.

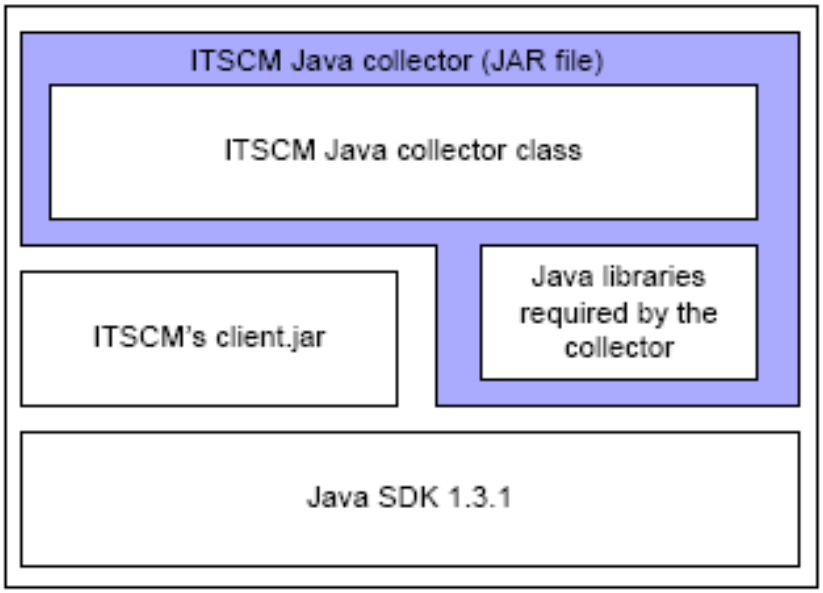

Abbildung 3.1: siehe [TSCM05] - TSCM Java Collector (JAR file)

Jede Instanz eines Collectors läuft in der Java Virtual Machine (JVM) des TSCM in einem eigenen Thread.

3.3.3 Tivoli Risk Manager (TRM)

Im Gegensatz zum TSCM, der nur Betriebssysteme überprüft, verwaltet der TRM Sicherheitsvorfälle, Schwachpunkte und Sicherheitsereignisse von Sicherheitsprodukten. Durch Korrelation von sicherheits-relevanten Informationen und Gefahrenmeldungen von Intrusion-Detection-Systemen, Firewalls, Virenscannern, Routern und Netzen ermöglicht der TRM eine zentrale Identifikation und Einstufung aller Bedrohungen und Angriffe auf ein lokales Netzwerk (LAN).

Auf Grund der zentralen Steuerung des TRM an einer Konsole können im Falle eines Angriffs Sicherheitsrichtlinien im ganzen System umgesetzt werden, um so dem Angriff entgegenzuwirken.

3.3.4 Unterschiede zwischen Event und State Monitoring

Tivoli Event Monitoring wird eingesetzt, um alle sicherheits-relevanten Ereignisse eine Betriebssystems zu sammeln. Diese werden mittels TCP oder UDP an den TRM von Tivoli

übermittelt und dort ausgewertet. Die Daten werden üblicherweise im Klartext als Syslog-ng[7] Format an den TRM übermittelt.

sicherheits-relevante Ereignisse:

- Login und Logout
- Sicherheitsverletzungen
- Spezifikationsprüfungen: resource manager profile und resource profile
- Aufrufe von sicherheits-relevanten Anwendungen
- Änderungen der Sicherheitsrichtlinien
- Änderungen der User Profile (Passwortänderungen etc.)

Im Gegensatz zum Event Monitoring, welches Ereignisse im betrachteten System in Echtzeit protokolliert, werden beim State Monitoring in längeren Zeitabständen einzelne Momentaufnahmen des ganzen Systems[8] gemacht und ausgewertet. Hierbei werden Informationen über die Einstellungen von bestimmten sicherheits-relevanten Systemparametern ausgelesen und an den TRM gesendet. Der TRM analysiert die Daten und prüft, ob alle Systemparameter den Sicherheitsrichtlinien entsprechen. Diese Informationen werden mit Hilfe der Tivoli Kollektoren aus den Betriebssystemen ausgelesen.

[7] Erweiterter Standard zur Übermittlung von Log-Meldungen. Ng steht hier für 'new generation'. Nähere Informationen zu dem Standard Syslog-ng sind unter http://www.balabit.com/products/syslog_ng/ zu finden.

[8] Das System wird hier als ein zu betrachtendes (monitoring) Objekt behandelt

4 Das IBM Betriebssystem z/VSE

4.1 Geschichte

Dieser Abschnitt gibt einen kurzen Überblick über die Geschichte und die Entstehung des Betriebssystems z/VSE. Details und Besonderheiten werden bei dieser Betrachtung nicht berücksichtigt.

Das heutige z/VSE ist das Resultat einer über 40 Jahre andauernden Entwicklung. Als erstes Betriebssystem für kleinere und Mittelstandskunden und als Alternative zu dem größeren OS/360 wurde 1964 für den Großrechner S/360 das Betriebssystem Disk Operating System/360 (DOS/360) entwickelt. Dieses System arbeitete 1965 mit drei Partitionen[1] (siehe Abbildung 4.1) und einer Hauptspeichergröße von 32 KB.

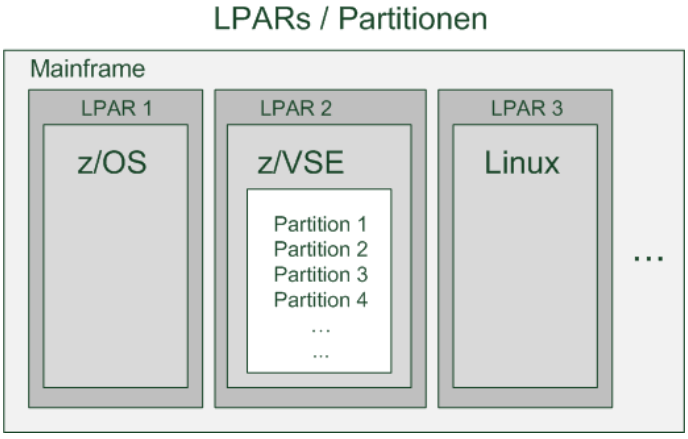

Abbildung 4.1: Unterschiede Mainframe LPARs und z/VSE Partitionen

[1] Eine Partition ist eine Unterteilung des virtuellen Adressraumes des Betriebssystems VSE in einzelne Abschnitte, die dann jeweils von laufenden Anwendungen belegt werden können.

Die Abbildung 4.2.2 veranschaulicht die Begriffe LPAR und Partition. Sowohl ein Mainframe LPAR als auch eine Partition im VSE werden als Partitionen bezeichnet. Der in diesem Kapitel verwendete Begriff Partition bezieht sich auf die Partitionen innerhalb eines VSE Systems.

Fünf Jahre später erschien eine neue Großrechnergeneration System/370 und 1972 wurden die Mainframes mit Virtual Storage erweitert, welches die Systemleistung erhöhte und die Programmierung vereinfachte. Zu diesem Zeitpunkt war der Speicher auf 256 KB angewachsen.

Das System DOS/360 durchlief 27 Releases bis es zu Disk Operating System/Virtual Storage (DOS/VS) umbenannt wurde. DOS/VS unterstützte fünf Partitionen und verbesserte durch einen Relocating Loader[2] die Effizienz der Multiprogrammierung.

Über die Jahre wurde der Speicher stetig erweitert und weitere Funktionen wurden hinzugefügt. Als ein neues Diskettensystem entwickelt wurde und sich der reale Speicher auf 4 MB belief, wurde das Betriebssystem Disk Operating System/Virtual Storage Extended (DOS/VSE) benannt. Ab diesem Zeitpunkt unterstützte DOS/VSE virtuellen Speicher (Virtual Storage Extended). Als virtueller Speicher (virtual storage) wird der Speicher bezeichnet, der tatsächlich von einem laufenden Programm physikalisch benutzt wird. Der virtuelle Speicher einer Anwendung, welcher im physikalischen Arbeitsspeicher völlig verteilt liegen kann, wird mit Hilfe der virtuellen Speicherverwaltung als logisch zusammenhängender Speicherbereich dargestellt. Die Abbildung der virtuellen auf die physikalischen Adressen wird vom Betriebssystem erledigt. Durch diesen Mechanismus belegt eine Anwendung nur so viel Speicher, wie es für die momentane Ausführung benötigt. Die restlichen, nicht im Speicher benötigten Daten werden solange auf einem permanenten Speichermedium abgelegt und bei Bedarf in den Speicher geladen. Die parallele Verarbeitung von Anwendungen wird durch die Bereitstellung mehrerer voneinander isolierter Adressräume (virtueller Speicherbereich) umgesetzt und optimiert.

DOS/VSE wurde 1984 in VSE/System Package (VSE/SP) umbenannt. Inzwischen war der Speicher auf 16 MB angewachsen. Etwa drei Jahre später 1987, wurde in die dritte Version von VSE/SP ein Komponentenkonzept implementiert, das bis heute im z/VSE verwendet wird. Die Struktur dieses Konzepts besteht aus einer Reihe an Basis- und optionalen Produkten, welche die Kunden je nach Anforderungen nachrüsten können.

Bereits 1990 wurde das VSE/SP zu VSE/ESA. Die erste Version hatte zum Ziel gute VSE/SP Konzepte zu übernehmen und den Fokus auf Qualität und Kapazitätssteigerung

[2] Relocating Loader: ermöglicht es ein Programm an einer beliebigen freien Stelle im Speicher abzulegen und auszuführen.

zu legen. Die Kapazität wurde gesteigert, indem der Adressraum des Betriebssystems von 24 Bit auf 32 Bit erhöht wurde. Jedoch konnten die 32 Bit nicht voll ausgeschöpft werden. Das erste Bit einer jeden Adresse wird verwendet um anzuzeigen, ob es sich um eine 32-Bit oder 24-Bit Adresse handelt. Wenn eine Adresse mit einer Null beginnt, handelt es sich um eine 24-Bit Adresse. Bei einer Eins am Anfang dreht es sich um eine 32-Bit Adresse, von der dann jedoch nur 31 Bit effektiv genutzt werden können. Der Grund für diese Unterscheidung in 24 und 31 Bit Adressen liegt in der geforderten Kompatibilität zu alten Assembler Programmen, die nicht neu kompiliert werden können. Durch die Erweiterung des Adressraums auf 31 effektive Bit konnten bis zu 2 GB im Speicher gehalten werden. Des Weiteren wurden in der neuen Version dynamische Partitionen umgesetzt, d.h. Partitionen sind nicht mehr auf ihre vorher festgelegte Größe beschränkt, sondern können sie je nach Anforderung anpassen.

Der CICS Transaction Server, welcher im nachfolgenden Kapitel 4.2.2 erklärt wird, wurde 1999 mit VSE/ESA V2.4 unterstützt.

2005 wurde aus VSE/ESA V2 das Betriebssystem z/VSE V3. Zu diesem Zeitpunkt war VSE noch nicht 64 Bit fähig. Dieser Mangel wurde im April 2006 behoben. Die neue Version z/VSE V4.1 aber unterstützt nunmehr die z-Architektur (Großrechner) und eine reale 64 Bit Adressierung.

4.2 Systemkomponenten

4.2.1 z/VSE 4.1 Central Functions

Die grundlegenden Funktionen des Betriebssystems z/VSE 4.1 sind:

- Speicherverwaltung
- Input / Output-Verwaltung der angeschlossenen Hardware
- Job Control Language (JCL)

Priority Output Writers, Execution Processors and Input Readers (POWER)

Diese Komponente ist das *spooling system* des VSE Betriebssystems. Ein *spooling system* verwaltet alle zu bearbeitenden Aufträge (Jobs) in einem Betriebssystem. Die anstehenden Aufträge werden in einem Puffer zwischengespeichert und mittels einer Stapelverarbeitung

dem ausführenden System oder Programm übergeben.

Funktionsumfang von POWER:

- Lesen und Speichern von Aufträgen, die von verschiedenen Eingabegeräten geschickt werden. Alle Aufträge werden in der *input queue*[3] abgelegt.
- Aufträge aus der *input queue* in einer Partition starten.
- Speicherung des Outputs, der von gelaufenen Jobs zurückgeliefert wurde, in der *output queue.*
- Jobs oder Output an Anwendungen schicken, die in anderen Partitionen laufen.

Virtual Storage Access Method (VSAM)

VSAM ist das Datenmangement- und Zugriffssystem für VSE Dateien. In VSAM-Dateien werden hoch sensible Kundendaten gehalten. Diese werden mit der Komponente VSAM verwaltet, gepflegt, geändert und gesichert. Der Zugriff auf den Inhalt der Dateien erfolgt wie der Zugriff auf virtuelle Speicher (siehe 4.1 auf Seite 27).

Interactive Computing and Control Facility (ICCF)

Über diese zentrale VSE Komponente lässt sich das Betriebssystem administrieren. Mit Hilfe des ICCFs werden sowohl Aufträge, die in einer Partition gestartet werden oder einen Batch-Job anstoßen, als auch Anwendungen entwickelt und in der ICCF Library gespeichert.

4.2.2 Customer Information Control System Transaction Server (CICS TS)

Der CICS TS ist ein Transaktionssystem, mit dem Transaktionen auf einem Datenbestand ausgeführt und Anwendungen entwickelt werden können. CICS verwaltet den parallelen Zugriff auf Daten und Dateien, sorgt für die Integrität der Daten und die Speicherverwaltung. Des Weiteren autorisiert CICS Benutzer und bestimmt, in welcher Reihenfolge die Jobs ausgeführt werden.

[3] input queue ist die Warteschlange in der die Aufträge auf ihre Verarbeitung warten.

Wie in Abbildung 4.2 zu sehen, kann man sich das CICS als eine zwischen dem Betriebssystem und der Anwendung liegende Schicht vorstellen, die eine Reihe von Funktionen bereitstellt, um Transaktionen zu bearbeiten. Die Anwendung kommuniziert mit dem CICS TS und dieser wiederum mit dem VSE, welches mit einem externen Terminal in Verbindung steht und die Daten über den CICS TS an die Applikation sendet.

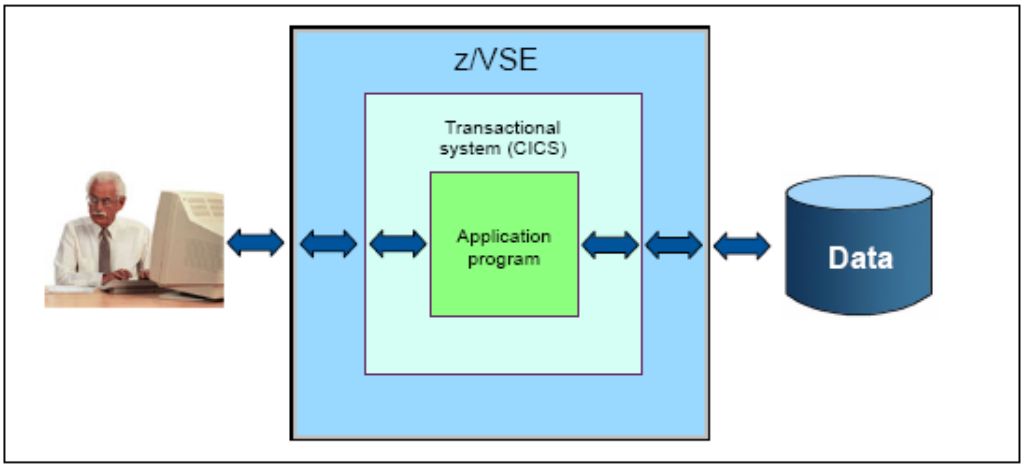

Abbildung 4.2: siehe [zVSEB06] - CICS Transaction Server und VSE

In einem VSE können mehrere CICS TS, jeder in seiner eigenen Partition, parallel laufen.

CICS TS unterstützt folgende Programmiersprachen:

- COBOL
- C
- C++ (nur z/OS nicht aber auf z/VSE)
- PL/I
- Assembler
- Java (nur z/OS nicht aber auf z/VSE)

4.2.3 TCP/IP for VSE

Die meisten Systeme benutzen inzwischen für die Kommunikation den Industriestandard Transmission Control Protocol/Internet Protocol (TCP/IP). Die U.S. Softwarefirma *CSI*

International[4] hat für z/VSE einen TCP/IP Stack entwickelt, der das Betriebssystem zu anderen TCP/IP-basierten Systemen kompatibel macht.

Einfach ausgedrückt bietet TCP/IP eine Reihe von Kommunikationsprotokollen an, die eine zuverlässige Ende-zu-Ende Verbindung zum Versenden von Daten gewährleisten. Weitere Kenntnisse von TCP/IP werden an dieser Stelle vom Leser vorausgesetzt.

Informationen zur Sicherheit von TCP/IP im VSE sind im Kapitel 5.6 auf Seite 52 zu finden.

Funktionalitäten von TCP/IP for VSE

- *File Transfer Protocol (FTP):* Dateien können mittels eines VSE Batch Programms und CICS transferiert werden oder das FTP wird direkt von einer VSE Applikation aufgerufen.
- *TN3270 (data stream):* Diese spezielle Variante von Telnet ist für Kommunikation zwischen dem VSE und den Terminals, über die mit dem VSE gearbeitet werden kann, zuständig. Die erweiterte Version TN3270E ist im RFC[5] 2355 definiert.
- *Line Printer Requester (LPR):* Mit diesem LPR Client ist es möglich, Mainframe Output oder Dateien auf im Netzwerk angeschlossenen Druckern zu drucken.
- *E-Mail:* TCP/IP for VSE bietet die Möglichkeit, E-Mails über das standardisierte SMTP Protokoll zu versenden.
- *General Printer Server (GPS):* GPS ermöglicht es auf VTAM-basierenden Programmen Aufträge an, über TCP/IP angeschlossenen Druckern, zu schicken ohne dass die Applikationen umgeschrieben werden müssen.
- *Network File System (NFS):* Dieses Software Paket ermöglicht anderen Computersystemen, die einen NFS Client installiert haben, auf entfernte VSE System Dateien zuzugreifen, als wären diese lokal verfügbar.

Um einen vollständigen Überlick zu erhalten werden die Systemkomponenten *DB2 Server for VSE* und *VTAM* an dieser Stelle nur kurz aufgeführt, da sie für die vorliegende Arbeit keine Relevanz haben.

[4] mehr Informationen unter http://www.e-vse.com/about-csi/about-csi.htm
[5] In RFCs werden Standards definiert. Weitere Informationen unter http://www.rfc-editor.org

4.2.4 DB2 Server for VSE

Der DB2 Server für z/VSE ist ein speziell für VSE und VM entwickeltes 'Relational Database Management System', das nicht mehr weiterentwickelt wird. Die Kunden von z/VSE sollen weg von dem DB2 Server für z/VSE und hin zu der Alternative 'Linux on System z' geführt werden. Das Linux, das auf z/VSE-Daten und Anwendungen zugreifen kann, soll parallel auf dem Mainframe installiert werden, um so auch die Nutzung anderer Datenbanksysteme zur Verfügung zu stellen.

Der DB2 Server unterstützt TCP/IP Zugriff, um von entfernten Rechnern auf Daten zugreifen zu können. Des Weiteren werden erweiterte *stored procedure, recovery, backup und archiving* Funktionen bereitgestellt.

4.2.5 VTAM (Virtual Telecommunications Access Method)

VTAM ist die frühere IBM proprietäre Netzwerkkomponente zum Verwalten des *System Network Architecture*[6] (SNA) und damit der Vorläufer von TCP/IP. VTAM diente als Schnittstelle, die die Kommunikation und Datenübertragung zwischen dem Benutzer und den Applikationen steuert. Folgende Aufgaben werden von VTAM in der Netzwerkkommunikation übernommen:

- Verbindungsaufbau und Sessionhandling
- Überwachung und Kontrolle von Ressourcen
- Bereitstellung eines Interfaces, das von Benutzern geschriebenen Programmen den Netzwerkzugriff erlaubt

4.3 VSE Kommunikation

Die zu erstellende Klassenbibliothek benötigt Java-basierten Zugriff auf VSE. Aus diesem Grund stellt der auf Java-basierte Connector, der diesen Zugriff ermöglicht, einen wesentlichen Bestandteil dieser Arbeit dar.

Wie jedes System benötigt auch VSE verschiedene Schnittstellen, über die eine Kommunikation mit anderen Systemen möglich wird. In diesem Fall wird diese Problematik mittels so genannten Connectors gelöst, die im VSE als Bindeglied fungieren, so dass es möglich ist mit Anwendungen auf anderen Plattformen zu kommunizieren. Im z/VSE

[6] weitere Informationen: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ibmsnaro.htm

dienen die Connectors als Verbindungsschnittstelle zwischen dem Betriebssystem und offenen Standards wie zum Beispiel Java Plattformen oder SOA (service-oriented architecture) Anwendungen.

Heutzutage behandeln die meisten modernen Betriebssysteme ihre Dateien als sequenziellen Datenstrom (Byte-Stream). Im Gegensatz dazu bestehen die meisten Files im VSE aus mehreren Records (VSAM). Diese beinhalten die Benutzerdaten und können eine fixe oder variable Länge haben. VSE Anwendungen arbeiten mit den Records und greifen für gewöhnlich nicht auf einzelne Bytes zu. Mehrere solcher Records können zu einem Block zusammengefasst werden. Zwar besteht die Möglichkeit, auf Records basierende VSE Dateien in einen sequenziellen Datenstrom umzuwandeln, jedoch gehen hierbei wichtige Informationen wie die Länge des Records verloren. Da Records oft aus Strings, binären Daten und weiteren Datentypen bestehen, muss man diese Daten erst in ein Format, das für den Empfänger lesbar ist, übersetzen und die richtige Zeichenkodierung (ASCII / EBCDIC) wählen. Durch diese heterogenen Datentypen innerhalb eines Records ist es nicht möglich, einen Record komplett als ASCII oder binären Datensatz zu übermitteln. Für die Aufgabe der Datenübersetzung und -umwandlung stehen im VSE verschiedene Connectors zur Verfügung, die unter Kenntnis von VSE Spezifikationen die Kommunikation und Transformation der Daten nach außen gewährleisten. Mit Hilfe dieser Schnittstellen ist es möglich, über das Transportprotokoll TCP/IP auf VSE Daten zuzugreifen.

Folgende z/VSE Connectors stehen zur Verfügung:

- Java-based connector
- VSAM redirector connector
- VSE script connector
- Simple Object Access Protocol[7] (SOAP) connector
- DB2-based connector

Im folgenden Kapitel werden die wichtigsten VSE Connectors kurz beschrieben.

4.3.1 Java-based connector

Dieser Connector wurde im Jahre 2000 in die Version VSE/ESA 2.5.0 integriert und ermöglicht es, über alle Arten von Java Programmen auf VSE Funktionen und Daten zuzu-

[7] standardisiertes, auf XML basierendes, weit verbreitetes Protokoll mit dem es Anwendungen möglich ist Informationen über das Internetprotokoll HTTP auszutauschen.

greifen. Dies beinhaltet z.B. auch Java Servlets oder EJBs, die in einer Web Application Umgebung laufen.

Client Teil (VSE Connector Client)

Der 'VSE Connector Client' kann auf den meisten java-fähigen Plattformen installiert werden. Der Zugriff auf VSE ist über die zwei Schnittstellen VSE Java Beans und VSAM JDBC Driver möglich. Des Weiteren wird die Kommunikation mit einem Web Application Server unterstützt.

Dieser Client-Teil wird im Folgenden von der zu erstellenden Security Klassenbibliothek verwendet werden.

Im Bild 4.3 wird veranschaulicht, wie sich eine Java Anwendung, die auf einem Server läuft, über die 'VSE connector client' Schnittstelle und TCP/IP zu dem 'VSE Connector Server' verbindet und auf VSE Daten zugreift.

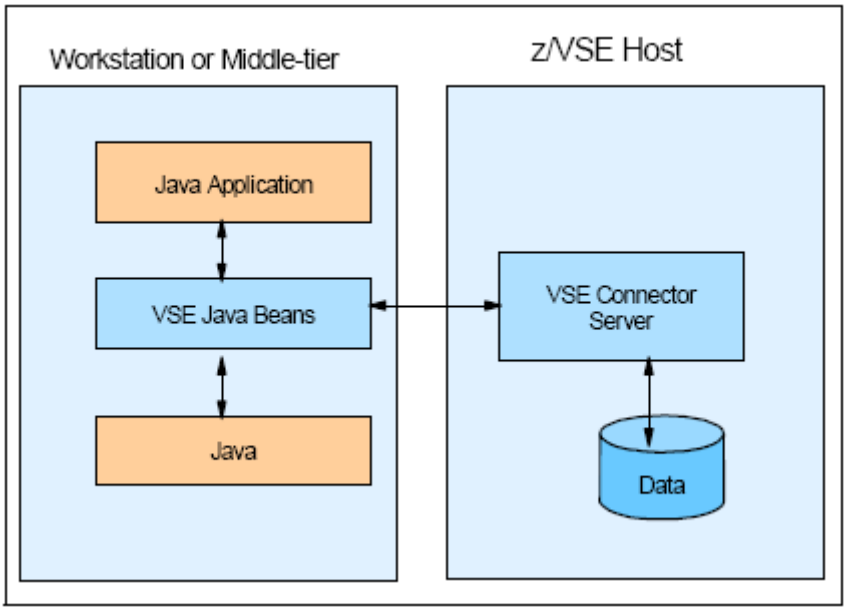

Abbildung 4.3: siehe [zVSEB06] - VSE Anbindung: Java-based Connector

Server Teil (VSE Connector Server)

Der 'VSE Connector Server' ist Teil des z/VSE Systems und läuft als Batch Anwendung. Ein Server kann mehrere Clients gleichzeitig bedienen.

4.3.2 VSAM Redirector Connector

Der VSAM Redirector Connector wurde Ende 2001 mit VSE/ESA 2.6 eingeführt. Mit diesem Connector ist es bestehenden VSE Anwendungen möglich, auf Dateisysteme und Datenbanken außerhalb von VSE zuzugreifen. COBOL Programme, die vor Jahrzehnten geschrieben wurden, können über diesen Connector ohne Veränderungen auf eine DB2 oder Oracle Datenbank zugreifen. Das Besondere hierbei ist, dass die angeforderten Daten von einer Java-basierten Plattform oder einem anderen Dateisystem von dem 'VSAM Redirector Server' im VSE Dateiformat zurückgeliefert werden, so dass die VSE Anwendung (COBOL) die Daten im richtigen Format und mit der korrekten Zeichenkodierung sofort verwenden kann. Die Anwendungen bekommen von der Umleitung nichts mit.

4.3.3 Simple Object Access Protocol (SOAP) Connector

Mit der im Jahre 2003 zur Verfügung gestellten Erweiterung des 'SOAP Connectors' ist das Betriebssystem z/VSE (erstmals VSE/ESA 2.7) in der Lage, mit allen Plattformen Informationen via Internet beliebig auszutauschen. Das VSE System kann mit Hilfe des Connectors als SOAP Server oder Client agieren. Hierbei läuft der eigentliche SOAP Server bzw. Client in der Transaction Server (CICS) Umgebung.

4.4 Datenspeicherung im z/VSE

Im z/VSE werden Informationen und Daten in verschiedenen Bibliotheken und Dateien abgelegt. In diesem Abschnitt werden die zwei unterschiedlichen Arten der Datenspeicherung betrachtet, soweit sie für diese Arbeit relevant sind.

VSE Dateien (Files)

- *Basic Access Method (BAM)-Dateien*
  Bei der Definition einer BAM-Datei muss die exakte Stelle, an der die Datei auf der Festplatte liegen soll, angegeben werden. Die Informationen über die Datei und deren Speicherort wird für die spätere Verwendung in einem Register (label area) abgelegt.

- *Virtual Storage Access Method (VSAM)-Dateien*
  Das Anlegen einer VSAM-Datei setzt einen definierten VSAM-Speicherbereich auf der Festplatte voraus, der in einem VSAM Katalog, welcher alle VSAM-Bereiche verwaltet, gespeichert wird. Nach diesen Schritten kann VSAM in besagtem Speicherbereich Platz für die VSAM-Dateien allokieren und Zugriffsmethoden bereitstellen. Der Umgang mit VSAM-Dateien ist durch die Zugriffsmethoden, die auch das Lesen und Schreiben organisieren, im Gegensatz zu den BAM-Dateien sehr viel komfortabler.

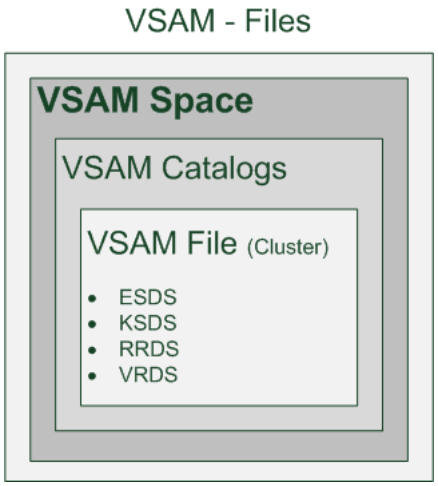

Abbildung 4.4: VSAM-Dateien

Die wesentlichen Merkmale der verschiedenen VSAM-Dateien bestehen in dem Zugriff und der Verwaltung. Man unterscheidet zwischen vier VSAM-Dateitypen.

  - Key Sequence Data Set (KSDS)
    Jeder Datensatz (kann eine beliebige Länge haben) ist mit einem Schlüssel versehen, mit dessen Hilfe es möglich ist, gezielt auf Datensätze zuzugreifen oder diese in die VSAM-Datei einzufügen.
  - Entry Sequence Data Set (ESDS)
    Bei dieser Form sind die Datensätze sequenziell geordnet.
  - Relative Record Data Set (RRDS)
    Die Datensätze sind durchnummeriert und in ihrer Länge beschränkt. Dadurch ist, wie bei KSDS, ein gezielter Zugriff möglich.
  - Variable Length Relative Record Data Set (VRDS)
    Die gleiche Methode wie RRDS nur mit variabler Länge der Datensätze.

Für die vorliegende Arbeit sind Zugriffe auf VSAM relevant, weil security-relevante Daten z.T. im VSAM abgelegt sind.

VSE Bibliotheken (Libraries)

Die VSE Bibliotheken können in einem VSAM-Speicherbereich oder ähnlich wie BAM-Dateien direkt auf der Festplatte abgelegt werden, jedoch wird ihre Baumstruktur: Libraries-Sublibraries-Members über eigene Methoden abgebildet.

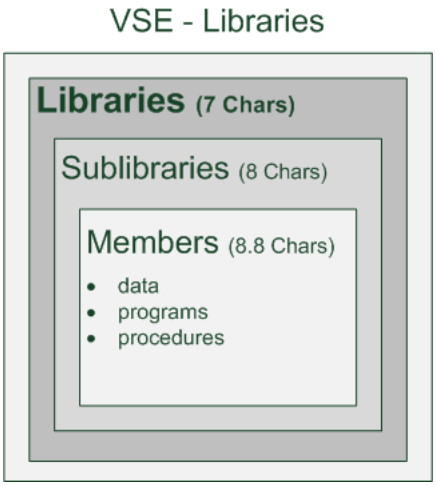

Abbildung 4.5: VSE-Bibliotheken

Die Members können Daten wie Programme, Prozeduren oder reinen Text beinhalten.

ICCF Bibliotheken (Libraries)

Hier stehen die ICCF Daten und Benutzerkennungen in einer Datei (DTSFILE), die wie eine BAM-Datei abgelegt ist. Die Bibliotheken, die sich wiederum aus Members zusammensetzen, werden über Nummern identifiziert. Die möglichen Inhalte von Members sind der Abbildung 4.6 zu entnehmen.

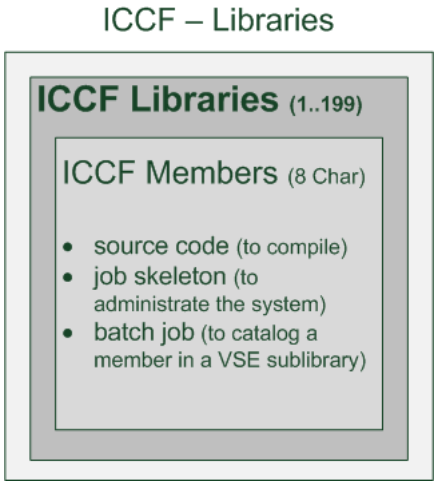

Abbildung 4.6: ICCF-Bibliotheken

Security-relevante Informationen werden teilweise als ICCF Member (DTSECTAB) abgelegt und sollen im Laufe dieser Arbeit ausgelesen werden.

5 Das VSE Sicherheitskonzept

Die Absicherung eines Betriebssystems vor firmeninternen und -externen Gefahren ist eine große Herausforderung und bedarf einer wohlüberlegten Strategie. Die Anforderungen an ein solches Sicherheitskonzept setzen sich aus verschiedenen Gesichtspunkten der IT-Sicherheit zusammen und reichen von Datenschutz bis hin zur gezielten Vergabe von Berechtigungen.
Das Sicherheitskonzept von VSE ermöglicht sehr feine Abstufungen der Sicherheitseinstellungen. Während des Starts des Systems wird entschieden, mit welcher Sicherheitsstufe, abhängig von den Anforderungen, das Betriebssystem hochfahren soll. Der Bootvorgang von VSE heißt IPL (initial program load). Hier wird entschieden, ob Online Security und/oder Batch Security aktiviert werden und welcher Security Manager verwendet werden soll.

Mittels eines Security Managers im VSE wird folgender Basisschutz bereitgestellt

- Kontrolle des Systemzugriffs der Benutzer
- Protokollierung der Aktivitäten eines autorisierten Benutzers auf Dateien und Programmen

5.1 System Authorization Facility (SAF)

Die System Authorization Facility (SAF Router) wird von den Ressourcen-Manager-Komponenten (CICS TS) benutzt, um sicherheits-relevante Entscheidungen wie Zugriffskontrollen zu treffen. Um den SAF Router zu verwenden, muss vorher das VSE Macro *RACROUTE* aufgerufen werden, welches den SAF Router aktiviert. Nach der Aktivierung ruft der Router den Basic Security Manager (BSM) oder einen External Security Manager (ESM) auf und beendet sich nach Übergabe der Verantwortung an den Security Manager von selbst.

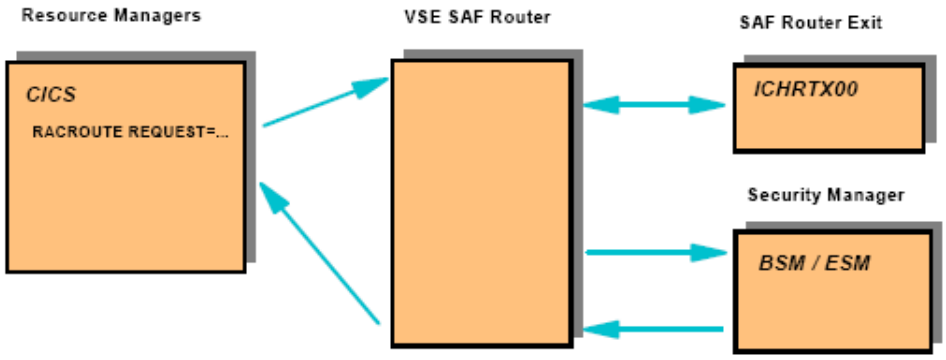

Abbildung 5.1: siehe [zOSB07] - System Authorization Facility

5.2 Basic Security Manager (BSM)

Der BSM ist Teil des z/VSE Betriebssystems und stellt die wichtigste Sicherheitskomponente dar. Die Hauptaufgaben des BSM liegen darin, Benutzer zu autorisieren und zu authentifizieren, die Datenintegrität zu erhalten und den Zugriff auf geschützte Systemressourcen zu verwalten und zu protokollieren. Hierfür hält der BSM alle Informationen über die Benutzer, die vorhandenen Ressourcen und Zugriffsdefinitionen (profiles) in einer Datenbank. Diese Daten werden aus drei unterschiedlichen Quellen eingelesen:

- VSE.CONTROL.FILE (VCF)
  Hier werden die Benutzerprofile in einem VSAM File gespeichert.
- BSM.CONTROL.FILE (BCF)
  Hier werden alle Ressourcen und die dazugehörigen Zugriffsrechte einer CICS Sitzung in einem VSAM File abgelegt.
- DTSECTAB Tabelle
  Hier werden die Sicherheitsdefinitionen für alle VSE files, libraries, sublibraries und members in einem ICCF Member festgelegt.

Auf diese gespeicherten Sicherheitsdaten greift der BSM zurück, wenn entschieden werden muss, ob ein Benutzer Zugriff auf einen geschützten Bereich erteilt bekommt oder nicht.

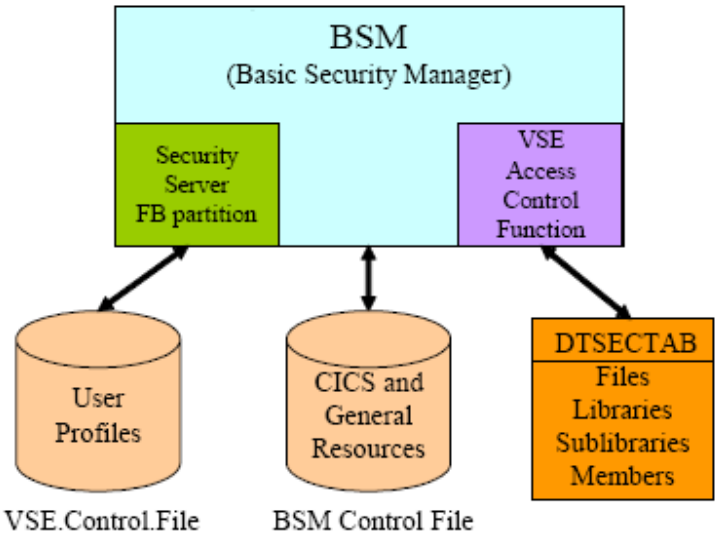

Abbildung 5.2: siehe [zOSB07] - Basic Security Manager-Daten

Die Art des Zugriffs auf geschützte Ressourcen (BCF) wird entweder durch einen direkten Eintrag des Benutzers in einem Ressourcen-Profil gesetzt oder durch einen Eintrag der Benutzerkennung in einer Gruppe. Die Gruppen müssen äquivalent zu den Benutzern einen direkten Eintrag in den Ressourcen-Profilen haben. Wenn ein Benutzer in einer Gruppe und gleichzeitig direkt im Ressourcen-Profil eingetragen ist, überschreibt der direkte Autorisierungseintrag des Benutzers die Gruppenrechte. Eine detailliertere Erklärung ist im Kapitel 5.4 auf Seite 51 zu finden.

Die Administration und Pflege der Einträge im BCF wird durch vom BSM bereitgestellte BSTADMIN Kommandos (siehe Abschnitt 5.2.1 auf Seite 48) ermöglicht.

Die Zugriffsrechte bei den DTSECTAB-Einträgen (files, libraries, sublibraries, members) werden entweder über eine allgemeine Berechtigung (universal access) oder über 32 Berechtigungsklassen verwaltet, welche ein granulareres Zugriffsmodell erlauben.

Bei der allgemeinen Berechtigung eines DETSECTAB-Eintrags (ausschließlich für libraries, sublibraries und members) wird nur festgelegt, welcher Zugriff allen Benutzern, die kein spezifischeres Recht haben, erteilt wird. Im Gegensatz dazu werden bei den Berechtigungsklassen in den Benutzerprofilen (siehe Anhang Benutzerverwaltung F.1 auf Seite 125) für jede Klasse die Rechte: `Connect, Read, Update, Alter` oder `No access` einzeln gesetzt.

Beispiel der 32 Berechtigungsklassen eines Benutzerprofils

```
Specify the access rights for 1-
32 DTSECTAB access control classes
( _=No access, 1=Connect, 2=Read, 3=Update, 4=Alter )
01 _ 02 3 03 3 04 3 05 3 06 _ 07 2 08 _ 09 1 10 _ 11 _
12 _ 13 _ 14 _ 15 _ 16 _ 17 _ 18 _ 19 _ 20 _ 21 _ 22 _
23 _ 24 _ 25 _ 26 _ 27 _ 28 _ 29 _ 30 _ 31 _ 32 _
```

Ein Benutzer hat nur dann den in seinem Profil gesetzten Zugriff (`Connect, Read, Update oder Alter`) auf einen DTSECTAB-Eintrag, wenn die definierte Klasse in dem jeweiligen Eintrag angegeben ist.

Beispiel eines DTSECTAB-Eintrag (library)

```
DTSECTAB TYPE=LIB,
NAME=888888.P.C.TEST.LIB1,
ACC=(09),
LOG=(1-8)
DTSECTAB TYPE=MEMBER,
NAME=LIB1.SUBA,
ACC=(03)
```

Im Anhang A.1 ist ein größerer DTSECTAB Auszug zu finden.

In diesem Beispiel hat der Benutzer auf die Library `888888.P.C.TEST.LIB1` das Zugriffsrecht `Connect` und auf das Member die Berechtigung `Update`. Die Zugriffskontrollfunktion richtet sich nach der VSE Bibliotheken Hierarchie (siehe Abbildung 4.5 auf Seite 38). Das heißt wenn eine Bibliothek geschützt ist, haben auch alle Unterbibliotheken und Dateien (Members) den gleichen Schutz. Jedoch können keine Unterbibliotheken und Dateien geschützt werden, wenn die Bibliothek darüber keinem Schutz unterliegt. Des Weiteren überschreibt das Recht des Elternobjekts immer das Recht der Kinderobjekte.

```
Sublibrary REP1.DEV..... access right UPD
Member     REP1.DEV.PROG1..... access right READ
```

Hier erbt das Member automatisch das Recht `Update` von der oberen Sublibrary.

Tabelle 5.1: Zugriffsrechte für Libraries, Sublibraries und Members

Access Right	**Library**	**Sublibrary**	**Member**
ALT	Create and delete.	Create, delete and rename	Create, delete and rename
UPD	Update contents. Create, delete and rename (ALT) sublibraries in it.	Update contents. Catalog, delete and rename (ALT) members in it.	Update contents. Add, delete and change lines.
READ	Read only for library and all sublibraries in it.	Read only for sublibrary and all members in it.	Read only.
CON	Access to sublibraries in it, if user has access right for these sublibraries individually.	Access to members in it, if user has access right for these members individually.	Not Applicable.

Die Definition `LOG=(1-8)` bedeutet, dass alle erfolgreichen Zugriffe der Klassen 1-8 auf diesen DTSECTAB-Eintrag mitprotokolliert werden. Unberechtigte Zugriffe werden automatisch registriert.

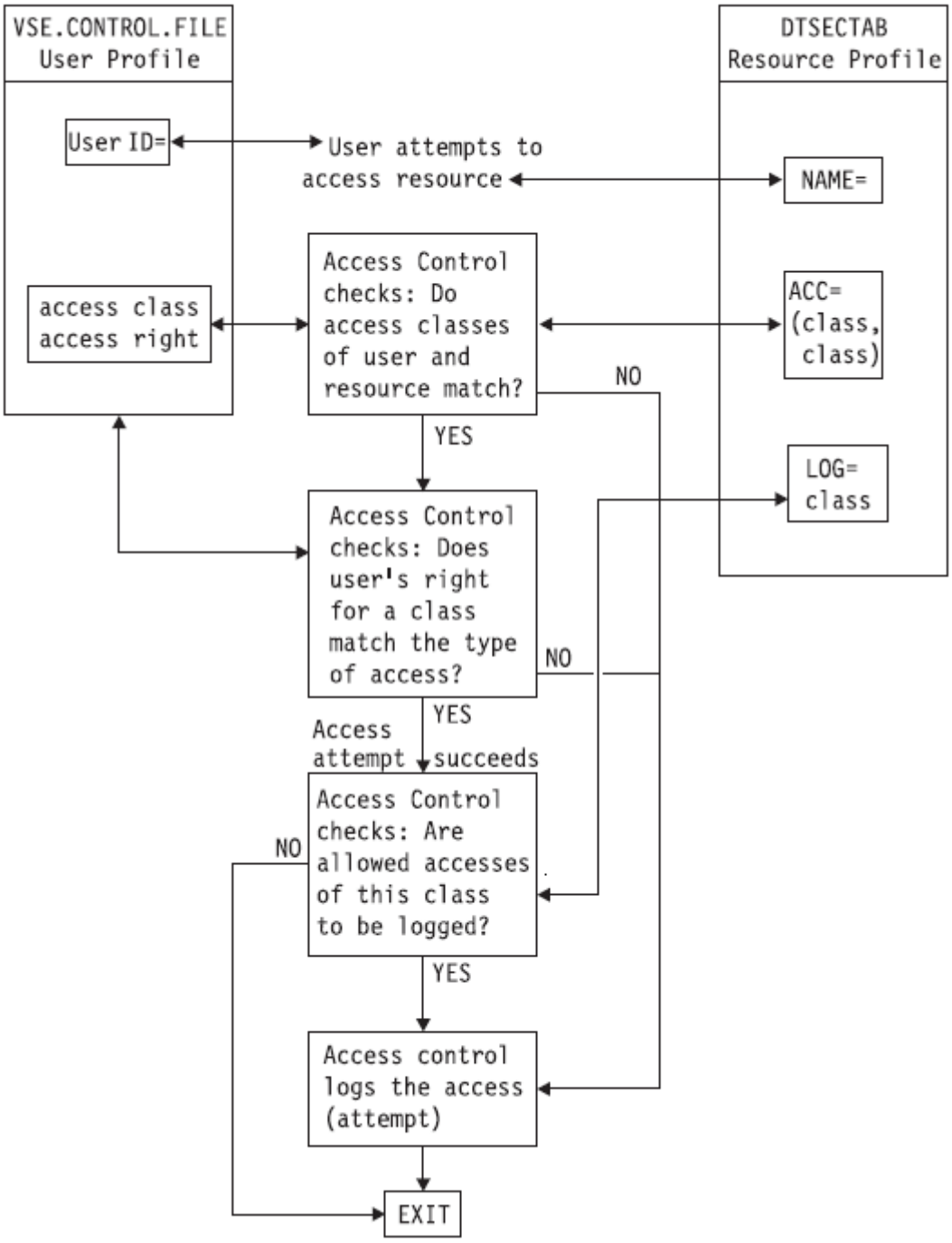

Abbildung 5.3: siehe [zVSEAd07] - Prüfung der Zugriffsautorisierung eines Benutzers auf einen DTSECTAB-Eintrag

In dieser Abbildung lässt sich nachvollziehen welche Schritte im BSM ablaufen, wenn ein Benutzer Zugriff auf eine Library, Sublibrary, File oder Member, deren Zugriffsrechte im DTSECATB definiert sind, fordert. Wenn in der DTSECAB kein *universal access* aktiviert ist werden die Berechtigungsklassen des Benutzers mit den DTSECTAB-Einträgen abgeglichen. Nicht genehmigte Zugriffe werden standardmäßig protokolliert. Wenn alle Zugriffe protokolliert werden sollen muss im DTSECTAB-Eintrag der Parameter `LOG` gesetzt sein.

Benutzer müssen sich über ihre Benutzerkennung und ihr Passwort als BSM-Benutzer identifizieren und authentifizieren. Mit Hilfe von Passwortrichtlinien legt der BSM fest welche Form sichere Passwörter haben und nach welchen Zeiträumen diese geändert werden müssen.

Zusätzlich werden mit der Überwachungsfunktion des BSM nicht autorisierte Zugriffe auf geschützte Ressourcen mitprotokolliert.

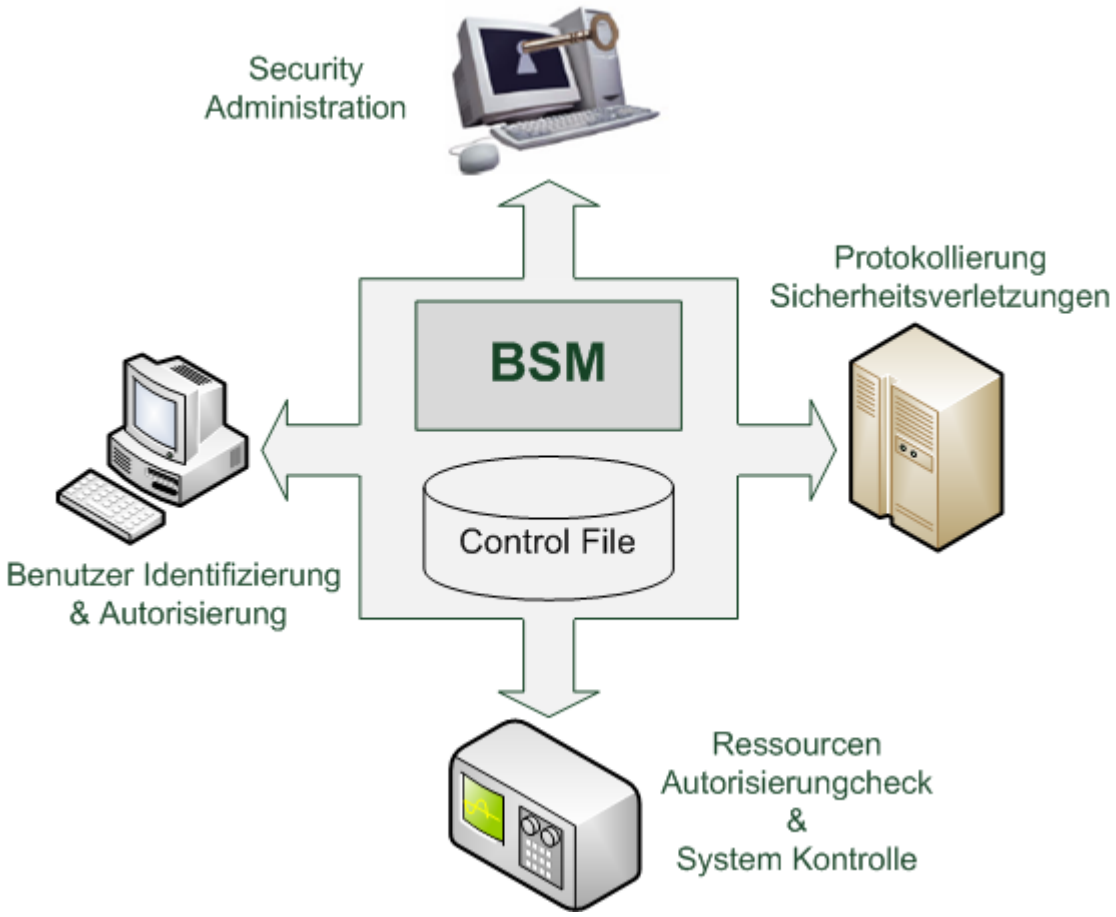

Abbildung 5.4: Basic Security Manager-Funktionsübersicht

5.2.1 BSM BSTADMIN Kommandos

Um das BSM.CONTROL.FILE (BCM) zu administrieren, stellt der BSM so genannte BSTADMIN Kommandos zur Verfügung, mit denen es möglich ist, alle Benutzer- und Gruppeneinträge detailliert zu verwalten.

Zusammenfassung BSTADMIN Kommandos

```
EXEC BSTADMIN
Commands:
```

```
ADD|AD class-name profile-name [GEN|NOGEN] [AUDIT(audit-
level)] [UACC(uacc)]
    [DATA('installation-data')]
CHANGE|CH class-name profile-name [GEN|NOGEN] [AUDIT(audit-
level)] [UACC(uacc)]
    [DATA('installation-data')]
DELETE|DE class-name profile-name [GEN|NOGEN]
PERMIT|PE class-name profile-
name [GEN|NOGEN] ID(name) ACCESS(access)|DELETE
ADDGROUP|AG group [DATA('installation-data')]
CHNGROUP|CG group [DATA('installation-data')]
DELGROUP|DG group
CONNECT|CO group user-id
REMOVE|RE group user-id
LIST|LI class-name profile-name|* [GEN|NOGEN]
LISTG|LG group-name|*
LISTU|LU user-id
PERFORM|PF [AUDIT ADMINACC|NOADMINACC] |
    [CLASS(class-name) ACTIVE|INACTIVE] |
    [DATASPACE REFRESH|SIZE(nK|nM)] |
    [PASSWORD [HISTORY|NOHISTORY]
        [LENGTH(minimum-pw-length)]
        [REVOKE(number-invalid-pws)|NOREVOKE]
        [WARNING(days-before-pw-expires)|NOWARNING]]
STATUS|ST
```

Mit den BSTADMIN LIST-Kommandos lassen sich Einträge mit bereits vergebenen Berechtigungen einsehen und auf der Konsole darstellen. Auf diese BCF-Beispieleinträge wird im Kapitel 8.1.2 auf Seite 69 näher eingegangen.

5.3 Abgrenzung von External Security Managers (ESM)

Um die Sicherheitseinstellungen im VSE zu verwalten und zu kontrollieren können anstatt des von IBM entwickelten *Basic Security Managers* (BSM) auch Security Manager von anderen Softwarefirmen eingesetzt werden. Im Folgenden werden die zwei wichtigsten *External Security Manager* (ESM) kurz vorgestellt und die jeweiligen Unterschiede zum BSM erläutert.

5.3.1 CA TopSecret

Der *External Security Manager* "CA TopSecret" wurde von der IT-Firma *Computer Associates*[1] entwickelt und stellt eine Alternative zum BSM dar. Die Funktionalitäten des CA TopSecret

- Benutzer Autorisierung/Authentifizierung
- Zugriffskontrolle auf Ressourcen
- Prüfung und Protokollierung von Zugriffen
- Garantie der Benutzerverantwortlichkeit
- SAF5.1 Kompatibilität

entsprechen denen des BSM. Jedoch hat der CA TopSecret im Gegensatz zum BSM eine benutzerorientierte Architektur. Das heißt die Sicherheitseinstellungen und Maßnahmen konzentrieren sich eher auf die Benutzer- als auf die Ressourcen-Konfiguration.

Der ESM von CA kann auf folgenden Systemen eingesetzt werden:

- z/OS
- z/VM
- z/VSE
- z/OS UNIX
- Linux for zSeries

5.3.2 BIM Alert/VSE

Diese Sicherheitskomponente für VSE wurde von der Firma B.I. Moyle entwickelt welche von CSI International[2] aufgekauft wurde. Im Vergleich zu den Security Managern BSM und CA TopSecret, die einen systemweiten Schutz anbieten, unterstützt der BIM Alert/VSE nur Batch (Job) Security. Der Security Manager überwacht und prüft die Benutzerkennungen, mit denen Batch Jobs gestartet werden. Abhängig von der Berechtigung des Benutzers, der den Auftrag (Job) gestartet hat, bekommt der Job von dem BIM Alert/VSE eine Sicherheitskennung. Anhand dieser Sicherheitskennung kann der Security Manager die Zugriffe auf geschützte Ressourcen verwalten und eventuelle Zugriffsverletzungen oder unerlaubte Programmaufrufe protokollieren.

[1] weitere Informationen zu CA sind unter http://www.ca.com/de/ zu finden
[2] weitere Informationen zu CSI International sind unter http://www.e-vse.com/about-csi/about-csi.htm zu finden

5.4 CICS Online Security

Alle Benutzer, die CICS Anwendungen betreiben, müssen sich mit einer Kennung identifizieren und einem Passwort am CICS System authentifizieren. Jeder dieser Anwender führt nun diverse Jobs und Aktionen aus, die alle in CICS Transaktionen umgesetzt werden. Dabei ist zu beachten, dass ein Benutzer nicht auf alle CICS Transaktionen und Ressourcen Zugriff hat, sondern ihm nur so viele Rechte eingeräumt werden, wie für seine Aktionen benötigt werden.
Um CICS Transaktionen auf Ressourcen zu schützen, werden dem BSM.CONTROL.FILE Einträge (Profile) mit den folgenden Angaben hinzugefügt:

BSM.CONTROL.FILE Profil

- Ressource Klassenname (bei Transaktionen immer TCICSTRN)
- Name der Ressource
- Zugriffsrecht

Bei dem Zugriffsrecht kann zwischen einem allgemeinem Zugriff (UACC = universal access), bei dem alle Benutzer das gleiche Recht (z.B. UACC(READ)) haben, und einer gezielten Rechtevergabe an Benutzer oder Benutzergruppen gewählt werden. Bei der gezielten Vergabe der Zugriffsrechte einzelner Benutzer oder Gruppen müssen diese der Zugriffsliste des Profils hinzugefügt werden.

Benutzerautorisierung für eine Transaktion:

1. BSM checkt den BSM Control File Eintrag, ob ein UACC (z.B. READ) definiert ist.
2. BSM prüft, ob der Benutzer auf der Zugriffsliste steht und ob das gesetzte Recht für den Zugriff ausreicht.
3. BSM prüft die Zugriffsliste auf einen Gruppeneintrag mit ausreichender Berechtigung und die Mitgliedschaft des Benutzers in dieser Gruppe.

Nur wenn einer der Punkte von eins bis drei erfüllt ist, wird der angeforderte Zugriff auf die Transaktion erlaubt. Der direkte Eintrag eines Benutzers in die Zugriffsliste eines Profils hat mehr Gewichtung als eine Zugriffsberechtigung, die über eine Gruppenmitgliedschaft an den Benutzer vererbt wird.

Allgemein verbietet CICS den Zugriff auf Transaktionen die nicht im BSM Control File eingetragen sind und damit keinem Schutz unterliegen.

5.5 Batch Security

Bei einem angemeldeten Benutzer wird ein Batch Job mit dessen Rechten ausgeführt. Wenn jedoch ein Batch Job ohne Anmeldung am System ausgeführt werden soll, muss in dem Job die Benutzeridentifizierung und Authentifizierung enthalten sein.

```
* $$ JOB ... SEC=(userid,password)
```

Ohne die Angaben eines Benutzers und Passwortes wird dem Job keine Berechtigung auf geschützte Ressourcen erteilt. Die Benutzerkennung und das Passwort werden vor dem Ausführen des Jobs durch den BSM geprüft. Der Batch Job läuft nach erfolgreicher Prüfung mit den Berechtigungen des im SEC Parameter angegebenen Benutzers. Dies schützt die Daten vor unerlaubten Zugriffen und Modifikationen.

5.6 TCP/IP Security im VSE

TCP/IP bringt neben den neuen Kommunikationsmöglichkeiten mit anderen Systemen auch eine große Menge an Sicherheitsproblemen mit sich. Da über dieses Protokoll die meisten Angriffsversuche unternommen werden, ist eine Absicherung des Systems von dieser Seite besonders wichtig.

Für TCP/IP Security müssen eigene Benutzerkennungen und Passwörter definiert werden. Diese werden dann im Klartext in einem Konfigurations-Member (Datei) *IPINITxx.L* abgelegt. Die Syntax für eine TCP/IP-Security Benutzerdefinition sieht wie folgt aus:

```
DEFINE USER,ID=id[,PASSWORD=pswd][,DATA=data],

[FTP=YES/NO,] Controls FTP access by this user

[LPR=YES/NO,] Controls LPR access by this user

[WEB=YES/NO,] Controls Web page access by this user

[TELNET=YES/NO,] Controls Telnet menu access by this user

[ROOT=directory] Restrict the userid to a specific directory
```

Wenn keine der Optionen (`FTP, LPR, WEB TELNET`) angegeben wird, sind diese standardmäßig erlaubt. Sobald aber eine dieser Optionen definiert ist, werden die Restlichen auf `NO` gesetzt.

Über einen *Security Exit (BSSTISX),* wie er in Abbildung *5.5* zu sehen ist, können die TCP/IP-Benutzerdefinitionen an den BSM übergeben werden. Hierbei werden die Benutzerkennung, das Passwort und die Zugriffsrechte auf Ressourcen oder Dateien überprüft und validiert. Wenn die Kontrolle nicht an den BSM übergeben wird oder übergeben werden kann, ist das System einzig durch die TCP/IP Sicherheitseinstellungen geschützt.

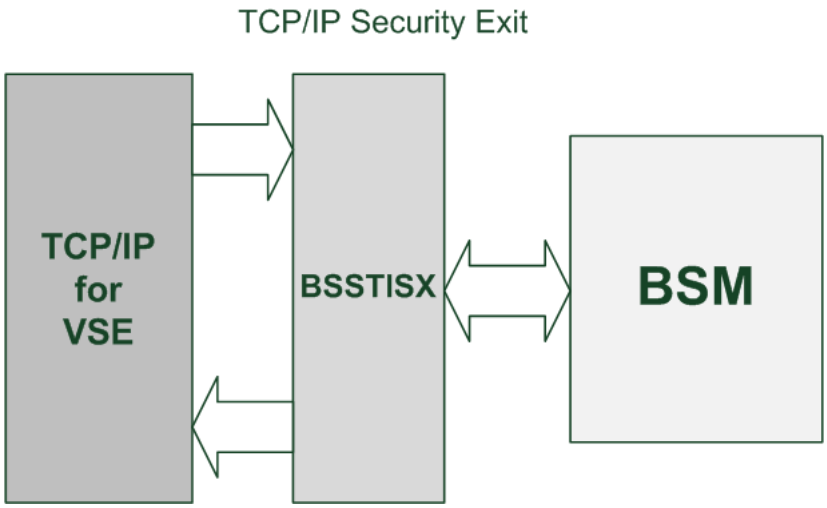

Abbildung 5.5: TCP/IP Security Exit

5.7 Connector Security

VSE Connectors, wie in Kapitel 4.3 beschrieben, kommunizieren in der Regel mit einem Server, der mehrere Endbenutzer bedient. Die Datensicherheit zwischen dem *VSE Connector Server* (VSE) und dem *VSE Connector Client* wird optional über eine SSL[3]-Verbindung (Secure Sockets Layer) gewährleistet. Diese SSL-Einstellungen werden in einer speziellen SSL-Konfigurationsdatei vorgenommen (siehe Anhang D.4 auf Seite 117).

Der *Connector Server* verhält sich wie ein Ressourcen-Manager. In jedem Fall muss sich der Client mit einer Kennung und einem Passwort anmelden. Der *Connector Server* verifiziert

[3] SSL ist ein Datenübertragungsprotokoll mit hybrider Verschlüsselung. Weitere Informationen dazu sind unter http://wp.netscape.com/eng/ssl3/ssl-toc.html zu finden.

die Anmeldung über ein RACROUTE-Aufruf bei dem *BSM* oder *ESM*. Nach erfolgreichem Login stellt der *Connector Server* über weitere Anfragen an den Security Manager fest, welche Zugriffsrechte der Client auf geschützte Systemressourcen hat.

Zusätzlich können Anfragen vom *VSE Connector Client* über Benutzerkennungs- oder IP-Adressen-Listen gefiltert und gesperrt werden (siehe Anhang D.1 auf Seite 115). Die Konfigurationsdateien *IESVCSRV.Z* (siehe Anhang D.3 auf Seite 116) und *IESLIBDF.Z* (siehe Anhang D.2 auf Seite 115) werden in VSE Bibliotheken gespeichert. In diesen werden allgemeine Verbindungs- und Sicherheitseinstellungen sowie Zugriffsrechte auf VSE Bibliotheken abgelegt.

5.8 Logon Security

Bevor sich ein Benutzer an dem VSE System mit seiner Kennung anmelden kann, muss diese Kennung im BSM in einem Benutzerprofil definiert sein. Die Benutzerprofile werden im VSAM in der VSE.CONTROL.FILE gehalten und legen fest, welche Rechte der Benutzer auf Ressourcen des Betriebssystems hat.

Des Weiteren werden hier die Gruppenzugehörigkeiten und die im Kapitel 5.2 erwähnten Berechtigungsklassen des Benutzers festgelegt. Ein als Administrator definierter Benutzer hat uneingeschränkten Zugriff zu allen geschützten Ressourcen.

6 Abgrenzung

Diese Arbeit und die Realisierung der VSESecurity Java Klassenbibliothek beziehen sich auf die internen sicherheits-relevanten Einstellungen und Parameter des z/VSE (Version 4.1 oder höher) Betriebssystems. Hierbei ist der BSM (siehe Kapitel 5.4 auf Seite 48) die wichtigste Quelle, da hier alle relevanten Sicherheitsinformationen verwaltet werden. ESM (siehe Abschnitt 5.3 auf Seite 49) von Drittanbietern werden bisher noch nicht unterstützt, können aber durch eine bereits vorgesehene Schnittstelle zur Klassenbibliothek hinzugefügt werden. Die VSESecurity Bibliothek stellt Daten folgender Bereiche zur Verfügung:

- Benutzereinstellungen
- Gruppenzugehörigkeiten
- Zugriffsberechtigungen von Benutzern und Gruppen
- Ressourcen-Sicherheit und Zugriff
- Zugriffsberechtigungen auf VSE Bibliotheken
- VSE Connector Sicherheit
- TCP/IP Sicherheit
- Systemeinstellungen und Sicherheit

Die gewonnenen Informationen über die Sicherheit des Betriebssystems können in Verbindung mit Anwendungen wie dem Tivoli TSCM (siehe Abschnitt 3.3.2 auf Seite 23) oder dem VSE Navigator[1] ausgewertet und überwacht werden.

Die Sicherheit des Großrechners selbst (Hardware Security), der permanenten Speichermedien (disk- oder tape-encryption) oder die des eventuell darüber liegendem Betriebssystems z/VM werden in dieser Arbeit nicht berücksichtigt. Des Weiteren werden Zusatzprodukte wie Firewall, Intrusion Detection und Antivirus Systeme, die typischerweise von Drittanbietern entwickelt werden nicht beleuchtet.

[1] weitere Informationen unter http://www.ibm.com/servers/eserver/zseries/zvse/products/connectors.html #navi

6.1 Vergleich von Sicherheitskonzepten

In diesem Abschnitt wird ein abstrahierter Vergleich der Sicherheitskonzepte verschiedener Betriebssysteme beschrieben. Dieser Vergleich soll eine grobe Übersicht von Sicherheitskonzepten geben und Unterschiede aufzeigen. Da Mainframe-Betriebssysteme die parallele Nutzung von mehreren Anwendern und Transaktionen verwalten müssen und PC-Betriebssysteme im Regelfall nur einige Benutzer, entstehen unterschiedliche sicherheitsrelevante Anforderungen an die Systeme. Aus diesem Grund lassen sich die Sicherheitskonzepte von Mainframe und PC-Betriebssystemen nur sehr eingeschränkt vergleichen. Darüber hinaus ist eine exakte Analyse aller Sicherheitskategorien der Betriebssysteme im Rahmen dieser Arbeit nicht möglich. Deshalb werden die Konzepte anhand weniger Kategorien verglichen und gegenübergestellt. Diese Kategorien werden im Folgenden kurz beschrieben.

- **Security Manager:** zentrale Verwaltung und Überwachung der Betriebssystemsicherheit
- **Resource Security:** Sicherheit und Schutz der Betriebssystem Ressourcen (Dateien, Ordner, Libraries etc.)
- **User Profiles:** Rolle eines Benutzers und die damit verbundenen Zugriffsberechtigungen
- **Authentification:** Benutzererkennung
- **Transaction Security:** Sicherheit von Transaktionen

Tabelle 6.1: Vergleich von Sicherheitskonzepten

	z/VSE	**z/OS**	**Linux**	**Windows**
Security Manager	BSM	RACF	Sicherheits - Funktionen im Betriebssystem integriert	Sicherheits - Funktionen im Betriebssystem integriert
Resource Security	**heterogenes Filesystem** • VSAM • ICCF • BAM • VSE Lib • POWER Spoolfile	**heterogenes Filesystem** • HFS • zFS • TFS • NFS	Dateien / Ordner	Dateien / Ordner Verschlüsselung von Ordnern
User Profiles / Roles	• Typ 1 Admin • Typ 2 Programmer • Typ 3 Gerneral	• Security Admin • Database Admin • System Programmer • System Operator • System Auditor	• root (Admin) • Benutzer	• Admin • Benutzer • Gast
Authentification	UserID Passwort (Secret)	UserID Passwort (Secret)	UserID Passwort (Secret)	UserID Passwort (Secret)
Transaction Security	Ja	Ja	-	-

Wie in dieser Übersicht zu sehen ist, haben nur die Systeme z/VSE und z/OS einen eigenen **Security Manager**, der die Systemsicherheit verwaltet. Der Schutz der System-Ressourcen (Dateien und Ordner) in den homogenen Filesystemen von Linux und Windows ist im Verhältnis zum Schutz der Mainframe-System-Ressourcen, die in heterogenen Filesystemen liegen, einfacher umzusetzen. Beispielsweise liegen die VSE Ressource-Zugriffsinformationen (BSTCNTL) in einem VSAM File (siehe Abschnitt 4.4 auf Seite 37) und die Zugriffsberechtigungen auf VSE libraries, sublibraries, files und members in einem ICCF Member (siehe Abschnitt 4.4 auf Seite 38), welche in assemblierte Form wiederum als VSE Library Member vorliegt.

Transaktionen und die dafür notwendigen Sicherheitsmaßnahmen werden nur von den Mainframe-Systemen unterstützt, da die Security Manager alle Zugriffe auf benötigte Ressourcen regeln. Eine Gemeinsamkeit aller Betriebssysteme ist das gleiche Authentifikationsprinzip über Benutzernamen und Passwörter. Des Weiteren können in jedem System Benutzer in Profile eingeteilt werden, welche diese grob in Berechtigungsklassen oder -rollen einteilt. Das Sicherheitsprinzip Divide and Conquer (zu deutsch "teile und herrsche") bei dem die Aufgaben und Berechtigungen auf die verschiedenen Benutzerrollen der Betriebssysteme aufgeteilt werden stellt einen wesentlichen Teil von Systemsicherheit dar. Hierbei sollte möglichst beachtet werden, dass Aktivitäten nicht von zwei Rollen ausgeführt werden können und dass die Person, die eine Aktion ausführt diese nicht bestätigen darf oder von dieser profitiert.

Resource Security

Das Sicherheitskonzept von VSE, welches bereits im Kapitel 5 behandelt wurde, ist dem des z/OS sehr ähnlich. Wie im VSE der BSM übernimmt auch im z/OS ein Security Manager (RACF) die Kontrolle und Überwachung der Betriebssystemsicherheit. Die wichtigsten Unterschiede liegen hierbei im Schutz der Ressourcen. Im Gegensatz zum VSE gibt es im z/OS hierarchische Sicherheitsstufen, die wie eine Baumstruktur aufgebaut sind. Alle Ressourcen sind mindestens einer Sicherheitsstufe zugeteilt. Sobald ein Benutzer eine Berechtigung für einen Knoten zugeteilt bekommen hat, gilt dieses Zugriffsrecht auf alle darunter liegenden Knoten. Dieses Konzept ermöglicht eine realitätsnahe Abbildung einer Firmenhierarchie. Außerdem bietet z/OS zusätzlich zur Protokollierung aller Zugriffe (Protokollierung wird auch von VSE unterstützt) die Funktionalität, den Besitzer einer Ressource bei einem Zugriff auf diese zu benachrichtigen.

Im VSE besteht die Möglichkeit den Security Manager mit dem Parameter `SYS SEC=NO` (siehe SIR[2] Kommando im Anhang B auf Seite 111) auszuschalten. In diesem Zustand werden nur noch die Berechtigungen der Benutzer beachtet und keine Ressourcen-Sicherheitseinstellungen. Diese Funktion lässt sich optimal für Testzwecke einsetzen.

Die Betriebssysteme Windows und Linux heben sich in vielen Punkten von den beiden Mainframe System z/VSE und z/OS ab. Allein der Schutz der Ressourcen in den homogenen Dateisystemen von Linux und Windows wird über die direkte Vergabe von Zugriffsberechtigungen auf Dateien und Ordnern geregelt. Des weiteren verfügen diese Betriebssysteme über keinen zentralen und austauschbaren Security Manager. Sicherheitsrelevante Funktionalitäten sind komplett in das Betriebssystem eingewebt und können nur durch zusätzliche Software erweitert werden.

Im Gegensatz dazu übernimmt bei den Mainframe-Systemen die Sicherheit des ganzen Systems ein einzelner Security Manager. Dieser Manager wird mit allen sicherheitsrelevanten System-, Benutzer-, und Ressourcen-Informationen versorgt. Durch diese zentrale Zugriffs- und Berechtigungsverwaltung ist es möglich, eine sichere Benutzung des Systems durch mehrere Anwender und Anwendungen zu gewährleisten. Hierbei kontrolliert und überwacht der Security Manager jede Art eines Zugriffs auf eine Ressource.

Weitere Unterscheidungsmerkmale von Mainframe und PC-Betriebssysteme liegen in Ansätzen und Anforderungen an die Sicherheitskonzepte. Auf PC Ebene spielt *Digital Right Management* (DRM) und das *Trusted Platform Module* (TPM), welches eine Maschine anhand einer festen Kennung identifiziert, eine große Rolle. Die Mainframe Systeme bleiben von solchen Themen momentan noch unberührt. Bei Mainframe-Sicherheit wird der Schwerpunkt auf Hardware Security gelegt. Hierbei werden Hardwaremodule verbaut, die komplett isoliert Daten ver- und entschlüsseln. Diese in Hardware gegossene Sicherheit macht das System nicht angreifbar, da sich beispielsweise diese Bauteile, bei Erkennung eines nicht regelkonformen Magnetfeldes[3] von selbst zerstören.

[2] mit dem SIR Kommando werden Systemeinstellungen gesetzt

[3] z.B. ist es möglich durch Messungen von Magnetfeldschwankungen einen Schlüssel der Verschlüsselung zu ermitteln.

7 Analyse

In diesem Kapitel werden die Analysephase des Entwicklungsprozesses und die Anforderungen an die Klassenbibliothek beschrieben. Im ersten Teil dieses Abschnitts werden die Mängel des Ist-Zustands (Abschnitt 7.1) erläutert. Daraufhin werden mit Hilfe des gewünschten Soll-Zustands (Abhscnitt 7.2) die Anforderungen (Abschnitt 7.3) an die Klassenbibliothek erarbeitet und festgelegt.

7.1 Ist-Zustand

Die Administration und Überwachung aller sicherheits-relevanten Parameter im VSE erfolgt über eine Benutzeroberfläche, die es dem Administrator ermöglicht, alle notwendigen Einstellungen bezüglich der System- und Zugriffssicherheit vorzunehmen. Jedoch ist eine automatisierte, regelmäßige Kontrolle dieser sicherheits-relevanten Parameter in einem akzeptablen Zeitfenster nicht möglich. Der Grund dafür liegt einerseits in der dezentralisierten Verwaltung und andererseits in der schwer überschaubaren Menge der zu kontrollierenden Sicherheitseinstellungen.

Mit von IBM entwickelten Anwendungen[1], wie dem 'VSE Health Checker' oder dem 'VSE Navigator', die dem Administrator ein GUI (Graphical User Interface) zur Verfügung stellen, können zwar einige system- und sicherheits-relevanten Daten eingesehen werden, jedoch ist es hier nicht möglich, alle Sicherheitsparameter zentral einzusehen und damit überwachen zu können. Allein der 'VSE Health Checker' bietet eine Möglichkeit, den Systemzustand (Performance und Auslastung) zu analysieren und kritische Parameter in Ampelfarben zu markieren. In diesem Kontext beschränkt sich die Überwachung des VSE Systems und wenigen Sicherheitseinstellungen auf eine manuelle Kontrolle seitens des Administrators.

[1] detaillierte Informationen über diese Anwendungen sind auf http://www.ibm.com/servers/eserver/zseries/zvse/downloads/ zu finden

7.2 Soll-Zustand

Die Komplexität und Einstellungen der Sicherheitsparameter, die nur durch aufwendige Administration und regelmäßigen manuellen Einsatz gepflegt werden können, sollen mit Hilfe des Tivoli TSCM (Kapitel 3.3.2 auf Seite 23) und State Monitoring überwacht werden. Hierfür soll der TSCM die in dieser Arbeit entwickelten Java Klassenbibliothek *VSESecurity* als Basis für einen vom Kunden entwickelten Collector verwenden um über die bereitgestellten Methoden auf die sicherheitspezifischen VSE Parameter zuzugreifen.

Durch die Anwendung der *VSESecurity* Bibliothek im Rahmen eines TSCM-Collectors und State Monitoring können die gewünschten Parameter regelmäßig und automatisch aus dem VSE Betriebssystem ausgelesen werden. Diese Daten werden dann dem Tivoli TRM (Kapitel 3.3.3 auf Seite 25) zur Auswertung übergeben, der diese mit den Sicherheitsrichtlinien vergleicht und auf Richtigkeit prüft.

Erst dann wird eine automatisierte, zentrale Überwachung aller Sicherheitseinstellungen des Systems und aller Zugriffsberechtigungen von Benutzern auf Ressourcen möglich sein.

In anderen Systemen existieren Ansätze, die die benötigten Daten über proprietäre Prozeduren sammeln, diese in eine Text-Datei schreiben und per FTP auf die Tivoli Plattform übertragen. Diese Lösung wurde jedoch für VSE nicht erwünscht.

7.3 Anforderungen

In diesem Abschnitt werden die Anforderungen an die Klassenbibliothek, deren Realisierung in dieser Arbeit beschrieben wird, festgehalten. Zunächst werden die Anforderungen (Requirements) eines VSE Kunden erläutert, die den Anstoß für diese Arbeit gegeben haben. Darauf folgend wird die Anforderungsliste erweitert, so dass das Resultat dieser Arbeit für alle VSE Kunden einen Mehrwert darstellt und eine umfassende Abdeckung aller sicherheits-relevanten Parameter gegeben ist.

7.3.1 Anforderungsliste

Anforderungen des Kunden

1. *State Monitoring* (siehe Abschnitt 3.3.4) von einigen System- und Sicherheitsparametern (z.B.: User Resource Access, User Access Rights, Security im SIR Output siehe Anhang B auf Seite 111) aus den VSE Quellen:

   a) SIR Systemkommando
   b) BSTCNTL Member mit BSTADMIN Kommando
   c) aktiven DTSECTAB Daten (DTSECTAB Phase im VSE)
   d) VSE.CONTROL.FILE

2. Verwendung des Java-based Connectors (siehe Abschnitt 4.3 auf Seite 35)
3. Verwendung der VSESystem Klassenbibliothek
4. Auswertung des *State Monitorings* über Tivoli Collectors (siehe Abschnitt 3.3.2.1 auf Seite 24)

Anforderungen von IBM

1. *State Monitoring* aller VSE Sicherheitsparameter und -einstellungen
2. Erstellung einer Java Klassenbibliothek (VSESecurity), die in den Tivoli TSCM (siehe Abschnitt 3.3.2 auf Seite 23) integriert werden kann
3. Einstellungen und Parameter sollen über getter-Methoden abrufbar sein
4. aussagekräftige Methoden- und Parameternamen
5. klar definierte Schnittstellen (Aufrufende der Klassenbibliothek haben nur Zugriff auf Schnittstellen, Internas bleiben verborgen)
6. Bereitstellung von ausgesuchten kombinierten Daten über *convenience* Methoden
7. Erstellung einer Javadoc mit englischen Kommentaren

Diese Anforderungen lassen sich ausformuliert in drei Bereiche unterteilen. Diese Klassifikation der Anforderungen folgt der Beschreibung in [KW03].

7.3.1.1 Geschäftsanforderungen (Business Requirements [BR])

1. [BR-1] Sicherheitseinstellungen und -parameter müssen in regelmäßigen Abständen automatisiert kontrolliert und überwacht werden können.
2. [BR-2] Zugriffsberechtigungen auf VSE Ressourcen müssen überschaubar sein.
3. [BR-3] Sicherheitseinstellungen und -parameter müssen zentral an einer Stelle eingesehen werden können.

7.3.1.2 Funktionale Anforderungen (Functional Requirements [FR])

[FR-1] Datenerfassung

Die Klassenbibliothek muss auf

1. [FR-1.1] Benutzerdaten (VSE.CONTROL.FILE) zugreifen
2. [FR-1.2] Zugriffsberechtigungsklassen (Benutzerprofile) von Benutzern zugreifen
3. [FR-1.3] Berechtigungsgruppen (BSM.CONTROL.FILE) zugreifen
4. [FR-1.4] Ressourcen-Berechtigungseinträge (BSM.CONTROL.FILE) zugreifen
5. [FR-1.5] VSE Libraries, Sublibraries, Members und Files (DTSECTAB) zugreifen
6. [FR-1.6] TCP/IP Konfigurationsdatei zugreifen
7. [FR-1.7] VSE Connector Server Konfigurationsdatei zugreifen
8. [FR-1.8] VSE Connector SSL Konfigurationsdatei zugreifen
9. [FR-1.9] VSE Connector Librarian Datei zugreifen
10. [FR-1.10] SIR Systemkommando Outputdaten zugreifen

[FR-2] Datenstruktur und Datenverarbeitung

Die gesammelten Daten müssen

1. [FR-2.1] zusammengeführt und bereitgestellt werden in einem oder mehreren
   a) [FR-2.1.1] Benutzer-Objekten
   b) [FR-2.1.2] Ressource-Objekten
   c) [FR-2.1.3] Gruppen-Objekten
   d) [FR-2.1.4] TCP/IP-Objekt
   e) [FR-2.1.5] Connector Server-Objekt
   f) [FR-2.1.6] Connector SSL-Objekt
   g) [FR-2.1.7] Connector Librarian-Objekt
   h) [FR-2.1.8] Systemkommando SIR-Objekt

   i) [FR-2.1.9] DTSECTAB-Objekt

2. [FR-2.2] untersucht und ausgewertet werden

   a) [FR-2.2.1] welcher Benutzer ist Mitglied in welcher Gruppe

   b) [FR-2.2.1] welcher Benutzer hat welche Berechtigungen auf welche Ressourcen

   c) [FR-2.2.1] welche Gruppe hat welche Berechtigung auf welche Ressource

   d) [FR-2.2.1] welcher Benutzer hat welchen Zugriff auf VSE Libraries, Sublibraries, Members und Files

   e) [FR-2.2.1] welche(r) Benutzer / IP-Adresse hat welche Zugangsberechtigung über den VSE Connector

7.3.1.3 Nicht-funktionale Anforderungen (Non-Functional Requirements [NFR])

Diese nach dem Standard ISO 9126 definierten Anforderungen beschreiben Qualitätsmerkmale wie Benutzbarkeit, Portabilität, Integrität, Zuverlässigkeit und Änderbarkeit. In diesem Zusammenhang werden nur einige Anforderungen genannt, da die detaillierte Ausarbeitung aller Qualitätsmerkmale den Rahmen dieser Arbeit sprengen würde.

1. [NFR-1] Die Klassenbibliothek VSESecurity soll intuitiv anwendbar sein
2. [NFR-2] Die Änderbarkeit und Skalierbarkeit der Klassenbibliothek soll gewährleistet sei

Die in diesem Kapitel aufgezählten Anforderungen werden in der Designphase (siehe Kapitel 8 auf Seite 67) berücksichtigt und im Kapitel 9 auf Seite 91 realisiert.

8 Design

Dieses Kapitel beschreibt das Design der entwickelten Java Klassenbibliothek und die Umsetzung der in Kapitel 7.3 definierten Anforderungen. Im Mittelpunkt der Designphase standen eine klar strukturierte Architektur im Sinne der Objektorientierung und die Einbettung in die bereits vorhandene Systemarchitektur. Des Weiteren mussten die verschiedenen Möglichkeiten, die VSE Sicherheitsdaten auszulesen (Kapitel 8.1), abgewägt und bewertet werden. Erst nach diesem Prozess folgten die Festlegung der Klassenstruktur und die Datenzusammenführung.

8.1 Datenzugriff und Erfassung

Um alle Sicherheitseinstellungen und Parameter aus dem VSE System auszulesen, muss die Java Bibliothek auf mehrere Datenquellen zugreifen und verschiedene Systemkommandos abschicken. Im Laufe der Designplanung stellte sich heraus, dass die Datenerfassung über verschiedene Wege erfolgen kann. Im Nachstehenden werden die unterschiedlichen Datenquellen und die Vor- und Nachteile der unterschiedlichen Datenerfassungsmöglichkeiten skizziert und erläutert.

Zu Beginn dieser Arbeit wurde überlegt, alle notwendigen sicherheitspezifischen Parameter mit Hilfe einer Anwendung, die auf dem VSE-Host läuft, zu sammeln und in Form einer Datei zur Verfügung zu stellen. Die Klassenbibliothek müsste bei diesem Lösungsansatz (LA 1) nur eine Datei einlesen und die gewünschten Parameter ausparsen. Jedoch erwies sich diese Lösung wegen des beträchtlichen Aufwands eine VSE-Anwendung zu programmieren und die Anwendung über einen entfernten Programmaufruf (*Remote Procedure Call*) zu starten, als ungeeignet. Ein weiterer Nachteil dieses Ansatzes (LA 1) wäre, dass das zu entwickelnde Plugin die Daten aus dem Speicher des Betriebssystems beziehen würde und nicht aus den unter Umständen bereits geänderten System- oder Konfigurationsdateien. Hierbei könnten veraltete Datenbestände das Ergebnis eines *State Monitoring*

(siehe Abschnitt 3.3.4) verfälschen. Darüber hinaus würde dieser erste Lösungsansatz eine eigene Partition des VSE Betriebssystems belegen, was nicht den Anforderungen der Kunden entspricht.

Deshalb wurde die Entscheidung getroffen, die Sicherheitsdaten aus den einzelnen VSE-Quellen zu beziehen und erst in der während dieser Arbeit entwickelten Java Klassenbibliothek zu zentralisieren. Das Auslesen dieser Daten kann über vier verschiedene Arten erfolgen, die im Folgenden bei jeder VSE-Quelle detailliert diskutiert werden und an dieser Stelle aufgeführt sind.

Mögliche Alternativen des Zugriffs auf diese VSE-Quellen sind:

1. Alternative: Direktes Auslesen der Daten aus dem VSAM File
2. Alternative: Verwendung eines System- oder eines BSTADMIN Kommandos (siehe Abschnitt 5.2.1 auf Seite 48) und parsen des Outputs
3. Alternative: Auslesen der Daten über ein Connector Server Plugin, das die Security-Daten über Funktionen zurückliefert.
4. Alternative: Konfigurationsdateien vom Host-System runterladen und den Inhalt parsen

Die folgenden Abschnitte zeigen die verwendeten Arten des detaillierten Zugriffs auf diese VSE Daten.

8.1.1 Benutzerprofile [VSE.CONTROL.FILE]

Die Benutzer eines VSE Betriebssystems und deren Einstellungen werden in dem VSAM-File VSE.CONTROL.FILE [VCF] definiert. Die Organisation der Datensätze im VCF ist KSDS (siehe Abschnitt 4.4 auf Seite 37). Da der Aufbau dieses Files fest definiert ist, bietet sich hier die Zugriffsalternative 1 an. Jedes Benutzerprofil des VCF soll einzeln als binärer Datensatz ausgelesen werden. Aus diesen binären Datensätzen sollen mit Hilfe von Bit-Schablonen die einzelnen Benutzereinstellungen ausgelesen werden. Diese Bit-Schablone orientiert sich an dem internen und streng vertraulichen Schema des VCF und definiert den exakten Beginn und die richtige Länge aller Profilparameter bitgenau. Der Vorteil dieser Zugriffsvariante liegt in der Flexibilität der Schablone an den Aufbau der Daten bezüglich möglicher Veränderungen in kommenden Versionen und der Unabhängigkeit des Inhalts.

8.1.2 Ressourcen und Gruppen [BSM.CONTROL.FILE]

In dem BSM.CONTROL.FILE [BCF] werden die Zugriffsberechtigungen von Benutzern und Gruppen auf Ressourcen und die Gruppenzugehörigkeiten von Benutzern abgelegt. Der Zugriff auf die Daten dieses VSAM-Files soll äquivalent zu dem beschriebenen Zugriff auf Benutzerprofile (VCF) im Abschnitt 8.1.1 folgen (Alternative 1). Da jedoch bei Gruppen und Ressourcen unterschiedliche Bit-Schablonen verwendet werden müssen ist bei jedem Datensatz eine Prüfung durchzuführen ob es sich um einen Gruppen- oder Ressourceneintrag handelt. Die Bit-Schablonen für Gruppen und Ressourcen müssen nach dem vertraulichen Format des BSF erstellt werden.

Das Parsen des BSTADMIN Konsolenoutputs (Alternative 2) wäre abhängig von der Art und Weise der Darstellung eines Datensatzes (siehe unten: Beispieleintrag für eine Ressource-Klasse). Diese Zugriffsweise wurde als ungeeignet eingestuft, da bei einer Veränderung dieser Darstellungsart die Syntax des Parsers jedesmal angepasst werden müsste.

In der BCF können Zugriffsberechtigungen für die folgenden Ressourcen-Klassen und Gruppen definiert werden:

- `GROUP01-GROUP64`
- `TCICSTRN (Transactions)`
- `MCICSPPT (Application Programms)`
- `FCICSFCT (Files)`
- `JCICSJCT (Journals)`
- `SCICSTST (Temporary Storage Queues)`
- `DCICSDCT (Transient Data Queues)`
- `ACICSPCT (Transactions-CICS START)`
- `APPL (Applications)`
- `FACILITY (Special Resources)`
- `DATASET (VSE files that have the format` *volid.fileid*`)`
- `USER (belongs to the User-Ids in the VCF)`
- `VSELIB (VSE libraries that have the format` *volid.fileid.libname*`)`
- `VSEMEM (VSE sublibrary members that have the format` *libname.sublibname.membername*`)`

- VSESLIB (VSE sublibraries that have the format *libname.sublibname*)

Ein Beispieleintrag für eine Ressource-Klasse:

```
BG 0000 CLASS    NAME
BG 0000 -----    ----
BG 0000 TCICSTRN IESN
BG 0000
BG 0000 UNIVERSAL ACCESS
BG 0000 ----------------
BG 0000 NONE
BG 0000
BG 0000 INSTALLATION DATA
BG 0000 -----------------
BG 0000 NONE
BG 0000
BG 0000 USER    ACCESS
BG 0000 ----    ------
BG 0000 GROUP61 READ
BG 0000 JIM     READ
BG 0000
BG 0000 BST904I RETURN CODE OF LIST IS 00
BG-0000 BST901A ENTER COMMAND OR EN
```

Wie hier zu erkennen ist, hat der Benutzer JIM und die Benutzer in der Gruppe GROUP61 die Zugriffsberechtigung READ auf die Ressource-Klasse TCICSTRN. Der UINIVERSAL ACCESS ist bei dieser Ressource deaktiviert.

Ein Beispieleintrag für eine Gruppe und deren Mitglieder:

```
BG 0000 INFORMATION FOR GROUP GROUP61
BG 0000 INSTALLATION DATA = NONE
BG 0000 USER(S)=
BG 0000 ANNA
BG 0000 BERTA
BG 0000 CICSUSER
BG 0000 HUGO
BG 0000 TONY
```

```
BG 0000 BST904I RETURN CODE OF LISTG IS 00
BG-0000 BST901A ENTER COMMAND OR EN
```

Die Ausgabe besteht aus dem Gruppennamen `GROUP61` und der darin enthaltenen Benutzer.

8.1.3 VSE Libraries, Sublibraries, Members und Files [DTSECTAB]

Die Zugriffsberechtigungen für VSE Libraries, Sublibraries und Members stehen in dem Member DTSECTAB (Auszug aus der DTSECTAB: siehe Anhang A.1 auf Seite 107). Diese Daten können in einem ICCFMember (siehe Abbildung 4.4 auf Seite 38) oder in einem Member einer VSE Library oder Sublibrary (siehe Abbildung 4.5 auf Seite 38) abgelegt werden. Dieses Member wird dann assembliert und als Phase in einer VSE Bibliothek gespeichert, da VSE Programme besser auf Phasen zugreifen können als auf andere Member-Typen.

Da mehrere DTSECTAB Members im System existieren können, aber nur ein Member aktiv sein kann (produktiv genutzt wird), besteht die Möglichkeit die DTSECTAB Daten aus dem Speicher des Systems abzurufen und auszulesen. Jedoch hätte auch hierfür eine VSE Anwendung geschrieben werden müssen, das die Daten sammelt und zurückliefert (siehe Lösungsansatz LA 1 im Abschnitt 8.1 auf Seite 67). In einer anderen Variante soll der Anwender der Java Klassenbibliothek in einer Konfigurationsdatei (vsesecurity.properties siehe Anhang C auf Seite 113) angeben können an welchem Ort die produktive DTSECTAB mit den aktuellen Daten im VSE abgelegt ist.

Der Zugriff auf die produktiven DTSECTAB Daten erfolgt in diesem Fall entweder über ein ICCFMember einer Library, über ein Member einer VSE (Sub)Library oder über eine lokale Datei. Der Inhalt des Members wird bei jeder dieser Möglichkeiten zuerst in ein Java-File-Objekt (Datei) geladen und dann zeilenweise ausgelesen (Alternative 4).

Durch eine Syntax- und Satzanalyse (String Parsing), die sich an der festgelegten Struktur des DTSECTAB Members orientieren wird, werden alle Parameter und Berechtigungen jedes Datensatzes (Libraries, Sublibraries, Members und Files) ermittelt und gespeichert. Der Aufbau der DTSECTAB ist dem Anhang A.1 auf Seite 107 zu entnehmen.
Ein DTSECTAB Beispieleintrag mit Erläuterung ist im Kapitel 5.2 zu finden.

In der DTSECTAB werden zusätzlich zwei Standardbenutzer (*FORSEC* und *DUMMY*) für Systemzwecke definiert. FORSEC hat den Status eines Systemadministrators und garantiert beim Systemstart gewisse Zugriffsrechte. Der Benutzer DUMMY hat keine speziellen Rechte und dient lediglich dazu, dass einige Prozeduren und Jobs, die zum Systemstart

notwendig sind, nicht mit den Administrationsrechten des FORSEC Benutzers gestartet werden müssen. Diese beiden Benutzer werden nur in der DTSECTAB definiert und dürfen nicht gelöscht werden. Die DTSECATB Benutzereinstellungen sollen auch ausgelesen werden.

8.1.4 VSE Connector (Server) Sicherheit [Connector Config Files]

Benutzer, die sich mit Hilfe des *VSE Connector Servers* (siehe Abschnitt 4.3) mit einem VSE verbinden und auf VSE Ressourcen zugreifen wollen, müssen sich davor am Betriebssystem anmelden. Jeder VSE Ressourcenzugriff über einen Connector wird geprüft. Die Berechtigungen sind hierbei von den Sicherheitseinstellungen des einzelnen Benutzers abhängig. Zusätzlich können in den Connector-Konfigurationsdateien (siehe Anhang auf Seite 115) Zugriffseinschränkungen abhängig von den Benutzerkennungen oder IP-Adressen und VSE Bibliotheken definiert werden (siehe Abschnitt 5.7).

Die Einstellungen des VSE Connector Servers können über ein *Connector Server Plugin*, das der Java Klassenbibliothek die Daten übermittelt, ausgelesen werden (Alternative 3). Bei dieser Methode müssen die Konfigurationsdateien des Connector Servers nicht geparst werden. Jedoch ist der Aufwand ein Server Plugin zu schreiben unverhältnismäßig groß und für diesen Zweck überdimensioniert.

Eine weitere Möglichkeit ruft die Connector Einstellungen über ein Systemkommando `STATUS` (Alternative 2) ab und parst den Output (`STATUS` Output siehe Anhang D.5 auf Seite 118). Jedoch liefert dieses Kommando keinen vollständigen Output aller Einstellungen. Beispielweise würden bei dieser Variante nicht alle SSL Parameter ausgegeben werden.

Aus diesen Gründen wurde in diesem Fall Alternative 4 gewählt. Hierbei werden die Connector-Konfigurationsdateien (VSELibraryMember), die in den VSE Libraries stehen, von dem System geladen und durch Syntaxanalyse ausgewertet.

8.1.5 TCP/IP Sicherheit [TCP/IP Config File]

Die Sicherheitseinstellungen für TCP/IP werden in einem Member (siehe Anhang E auf Seite 121) definiert. Das Auslesen der Daten erfolgt äquivalent zu den anderen Konfigurationsdateien (Alternative 4). Da aber der Administrator mehrere TCP/IP-Konfigurationsdateien anlegen kann, muss dieser den Ablageort des aktiven Members der VSESecurity Bibliothek mitteilen (siehe Anhang C auf Seite 113), so dass die richtigen Daten aus dem System extrahiert werden können.

Auch in diesem Fall hätten die Informationen über das Absetzen des Systemkommando `Q SET` (Alternative 2) aus dem System geholt und geparst werden können. Jedoch ist der Output wie in Abschnitt 8.1.4 unvollständig und deshalb für die Auswertung aller TCP/IP-Sicherheitsinformationen ungeeignet.

8.1.6 Systeminformationen

Allgemeinere Informationen über die Sicherheit und den Zustand des Systems erhält man im VSE über das Konsolenkommando `SIR`. Mit diesem Befehl bringt man beispielsweise in Erfahrung ob und wenn welcher Security Manager aktiv und welche Sicherheitsstufe im VSE eingestellt ist.

```
AR 0015 SEC. MGR. = BASIC SECURITY = ONLINE
```

In diesem Fall ist der BSM aktiv und Online Security eingestellt.

Einen kompletten Auszug des *SIR-Commands* ist im Anhang B auf Seite 111 zu finden. Der Output dieses Systemkommandos soll durch einen einfachen Parser ausgewertet werden (Alternative 2).

8.2 Klassenstruktur

8.2.1 Übersicht

Der Einstieg in die Klassenbibliothek erfolgt über die Klasse `VSESecurity` im Package *com.ibm.vse.security.* Wie in Abbildung 8.1 zu sehen ist, greift dieses Package auf alle anderen Packages zu. Die Klasse VSESecurity ruft die entsprechenden Klassen auf, die die einzelnen Datenquellen im VSE System ansprechen und die Daten zurückliefern.

Die folgenden Abschnitte liefern detaillierte Informationen über die Struktur und das Verhalten der beteiligten Klassen und Packages der Java Klassenbibliothek *VSESecurity.* Die hier gezeigten UML Diagramme zeigen nicht alle Methoden und Parameter,so dass eine gewisse Übersichtlichkeit gewahrt bleibt. Beschreibungen zu den einzelnen Methoden sind der Javadoc zu entnehmen, die sich auf der beigelegten CD befindet.

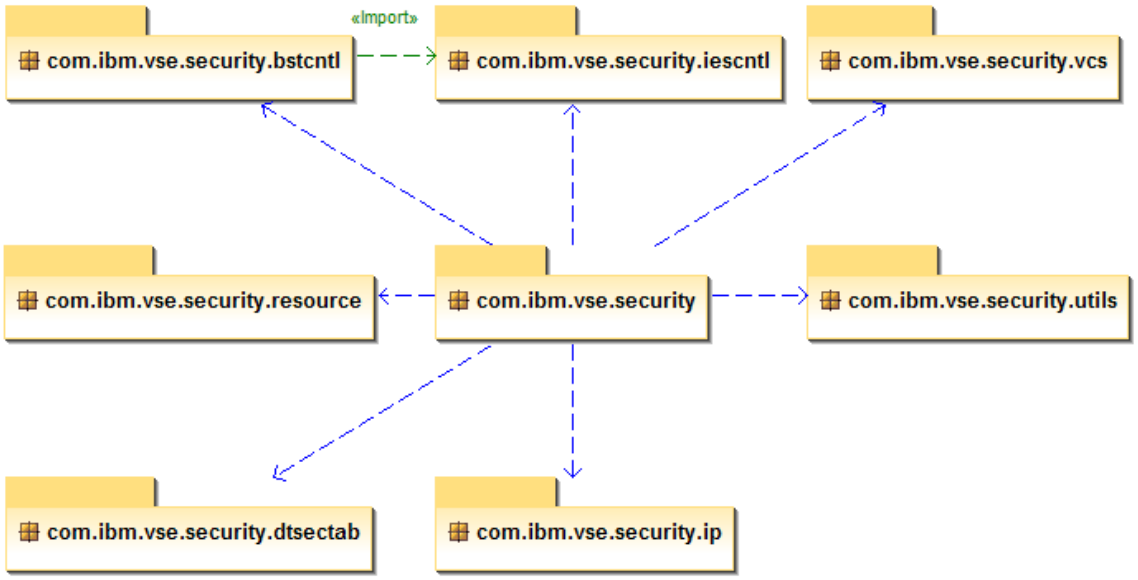

Abbildung 8.1: VSESecurity Packages Overview

8.2.2 Security Package

Das Security Package (*com.ibm.vse.security* siehe Abbildung 8.2 auf der nächsten Seite) ist die zentrale Stelle, an der alle Daten zusammenlaufen und über getter-Methoden abgerufen werden können.

Klasse VSESecurity

Diese Klasse ist der Einstiegspunkt für Anwendungen wie z.B. einen Tivoli Collector (siehe Abschnitt 3.3.2.1 auf Seite 24). Über den Konstruktor wird der Klasse ein VSESystem[1] Objekt übergeben, das ein VSE System repräsentiert. Die Sicherheitsdaten werden dann mit Hilfe der Klassen in den anderen Packages (siehe Abbildung 8.1) aus dem VSE ausgelesen und an dieser Stelle zum Abruf angeboten.

Nachdem die Daten komplett sind, werden sie von der Klasse `PrepareMergedData` so ausgewertet und zusammengeführt, dass zum Beispiel alle benutzerbezogenen Parameter über ein Benutzerobjekt `UserDetails` angesprochen werden können. Nach dieser Zentralisierung alle sicherheits-relevanten Daten ist es möglich alle relevanten Sicherheits-

[1] VSESystem ist eine Klasse aus der Bibliothek des VSE Connector Clients, die in dieser Arbeit verwendet wird

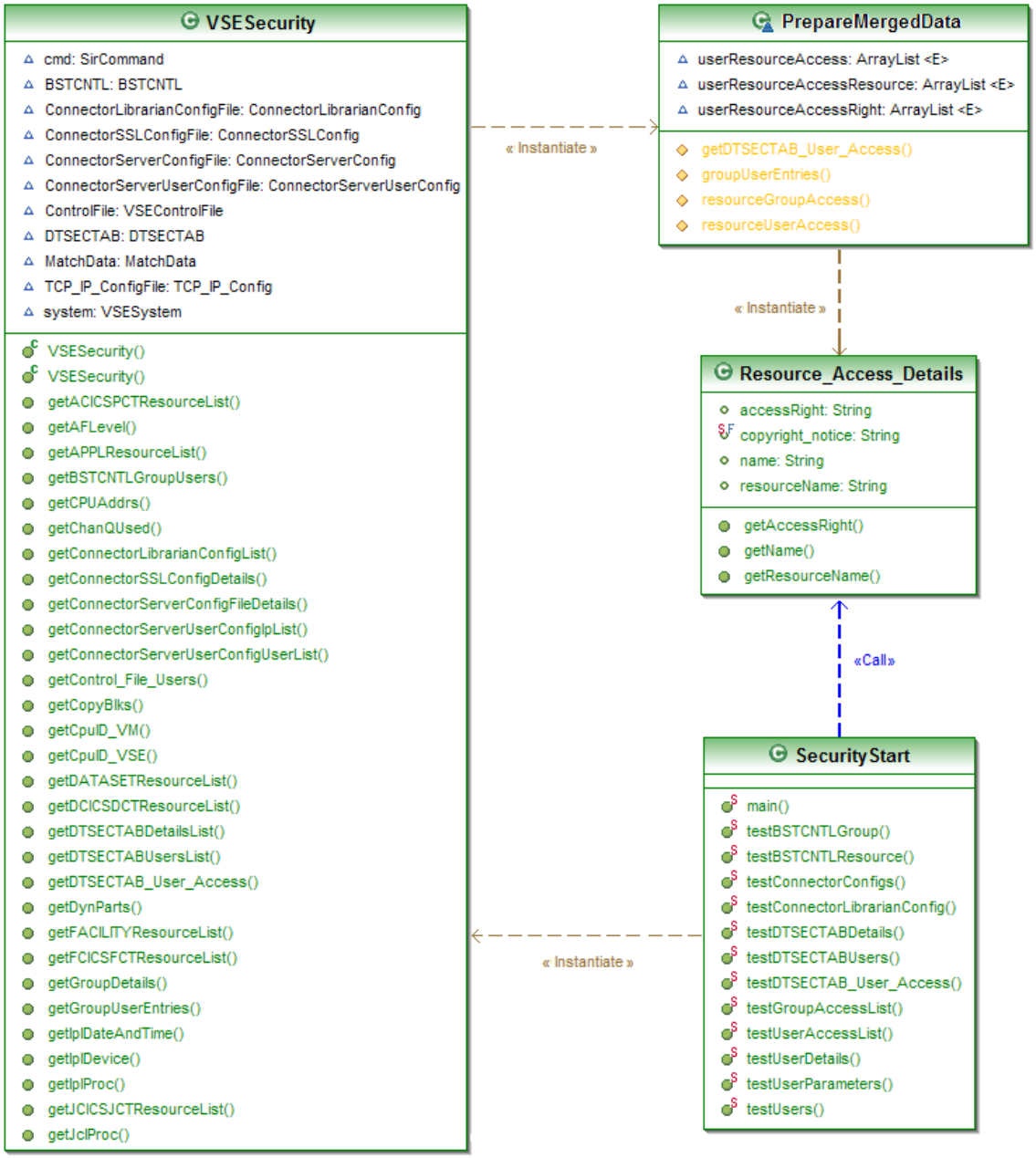

Abbildung 8.2: *com.ibm.vse.security* Package

einstellungen und Zugriffsberechtigungen über die Klasse `VSESecurity` einzusehen und zu kontrollieren.

Klasse Resource_Access_Details

`Resource_Access_Details` bietet die Möglichkeit vereinfacht Berechtigungseinträge auf Ressourcen darzustellen und dient zusätzlich als Hilfsobjekt für die Datenzusammenführungsklasse `PrepareMergedData`.

Klasse SecurityStart

Diese Klasse ist für den Test der Java Klassenbibliothek Funktionalitäten erstellt worden.

Klasse PrepareMergedData

Wie bereits in der VSESecurity Klassenbeschreibung erwähnt, ist die Klasse `PrepareMergedData` zuständig für die Auswertung und Zusammenführung der Daten. Hierbei werden die Benutzer-, Ressourcen-, Gruppen- und Bibliothekszugriffsdaten gegenübergestellt und verglichen. Mit dieser Prüfung wird festgestellt, welcher Benutzer in welcher Gruppe Mitglied ist und welchen Zugriff ein Benutzer auf Ressourcen und Systembibliotheken hat. Des Weiteren wird aus Sicht einer Ressource getestet, welche Gruppen oder Benutzer direkte oder indirekte Berechtigungen haben. Die Ergebnisse dieser Prüfungen werden dann den jeweiligen Objekten (Benutzer, Gruppen oder Ressourcen) nachträglich mitgeteilt, so dass alle Informationen über ein Objekt abrufbar werden.

8.2.3 Benutzer Package

In diesem Package (*com.ibm.vse.security.iescntl* siehe Abbildung 8.3 auf der nächsten Seite) sind alle Klassen enthalten, die für das Auslesen der Benutzerdaten zuständig sind.

Klasse VSEControlFile

Diese Klasse greift auf das VSE.CONTROL.FILE [VCF] zu und liest alle Benutzerdaten aus. Hierbei werden die einzelnen Benutzerdetails mit Hilfe der bereits erwähnten Bit-Schablone aus der VCF extrahiert und in Instanzen von der Klasse `Control_File_User_Details` gespeichert.

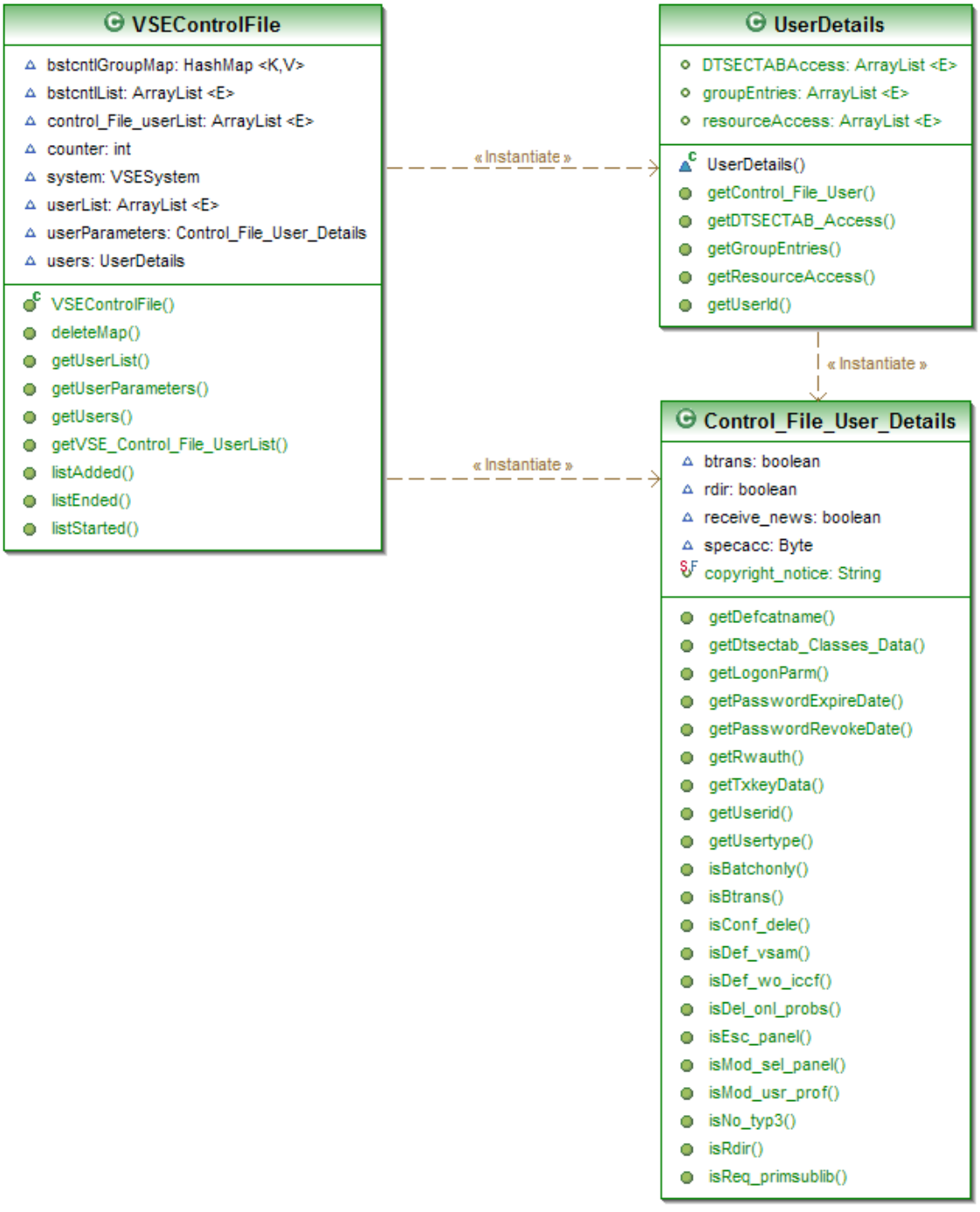

Abbildung 8.3: *com.ibm.vse.security.iescntl* Package

Klasse Control_File_User_Details

`Control_File_User_Details` ist eine Repräsentation aller Benutzerdetails im VSE Betriebssystem.

Klasse UserDetails

`UserDetails` stellt einige Methoden zur Verfügung, mit denen es möglich ist zusammengeführte sicherheits-relevante Daten eines Benutzers anzeigen zu lassen. Diese Daten stammen aus den verschiedenen Quellen des Systems und werden an dieser Stelle zentralisiert bereitgestellt.

8.2.4 Ressourcen und Gruppen Package

Mit diesen Klassen (siehe Abbildung 8.4 auf der nächsten Seite) werden die Gruppenzugehörigkeiten und Ressourcenberechtigungen der Benutzer im System bestimmt.

Klasse BSTCNTL

Über diese Klasse wird auf das BSM.CONTROL.FILE [BCF] zugegriffen. Die Klasse instanziiert die instanziiertKlassen `BSTCNTLResourceDetails, BSTCNTLGroupDetails, BSTCNTLAccessLists` und alle `Resource-Klassen.` Die Gruppen- und Ressourceneinträge und deren Berechtigungen werden als Listen zur Verfügung gestellt.

Klasse BSTCNTLAccessLists

Diese Klasse repräsentiert die Berechtigungseinträge einer Ressource. In einer Berechtigungsliste einer Ressource können sowohl Gruppen als auch Benutzer stehen. Diese Listen sind eines der wichtigsten Bausteine des VSE Sicherheitsprinzips.

Klasse BSTCNTLGroupDetails

`BSTCNTLGroupDetails` repräsentiert einen Gruppeneintrag im BCF.

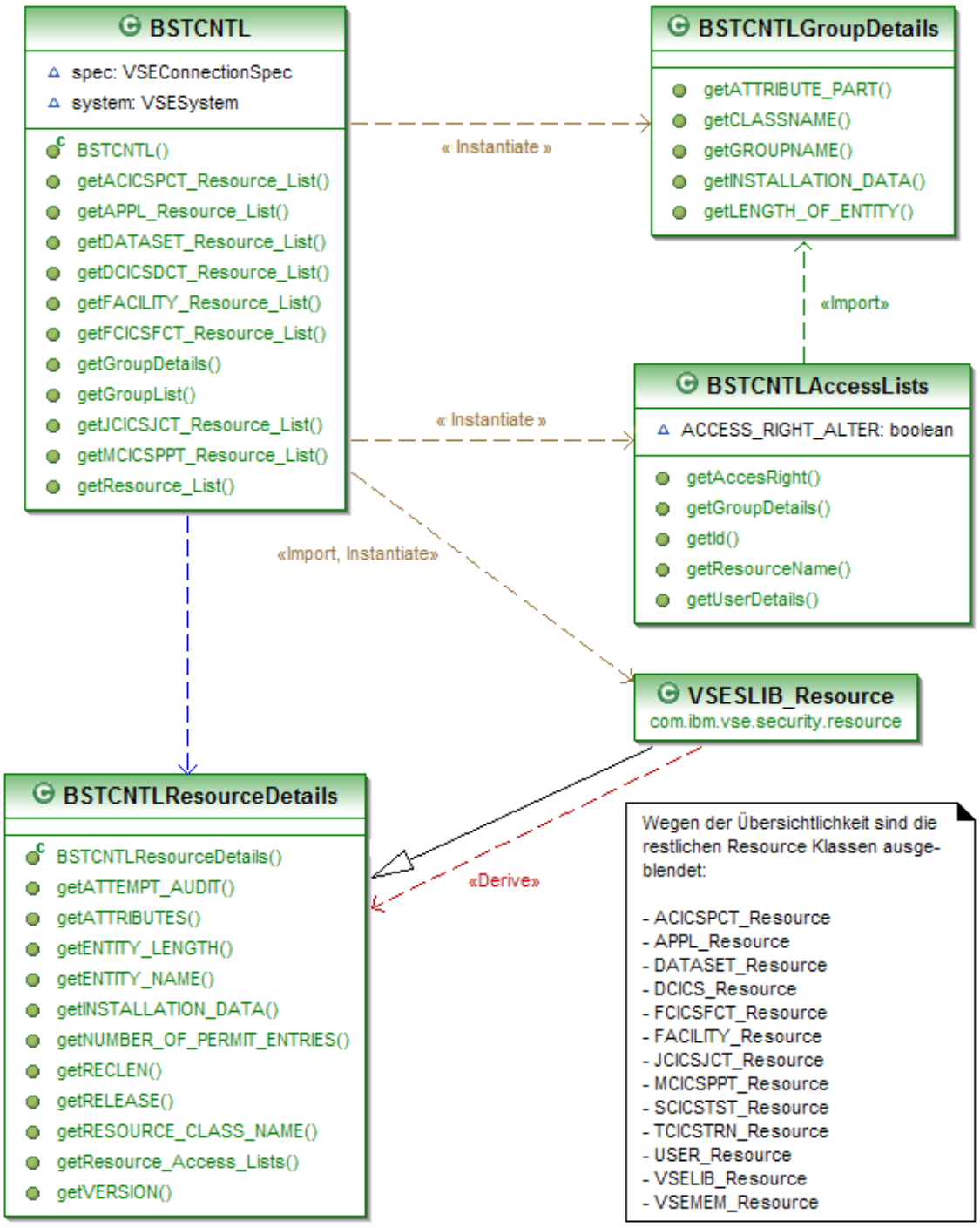

Abbildung 8.4: *com.ibm.vse.security.bstcntl* Package

Klasse BSTCNTLResourceDetails

`BSTCNTLResourceDetails` repräsentiert einen Ressourceneintrag im BCF. Über die Methode `getResource_Access_Lists()` erhält man beispielsweise eine Liste des Typs `BSTCNTLAccessLists` mit allen Berechtigungseinträgen einer Ressource.

Resource-Klassen

Die 14 Ressource-Klassen (siehe Abbildung 8.5 auf der nächsten Seite)

- `ACICSPCT_Resource`
- `APPL_Resource`
- `DATASET_Resource`
- `DCICSDCT_Resource`
- `FACILITY_Resource`
- `FCICSFCT_Resource`
- `JCICSJCT_Resource`
- `MCICSPPT_Resource`
- `SCICSTST_Resource`
- `TCICSTRN_Resource`
- `USER_Resource`
- `VSELIB_Resource`
- `VSEMEM_Resource`
- `VSESLIB_Resource`

sind Subklassen, die alle von der Superklasse (siehe Abbildung 8.5 auf der nächsten Seite) `BSTCNTLResourceDetails` erben, da alle der selben Struktur zu Grunde liegen.

8.2.5 VSE Libraries Packages

Diese Klassen (siehe Abbildung 8.6 auf Seite 82)werten die Zugriffsberechtigungen auf Systembibliotheken aus.

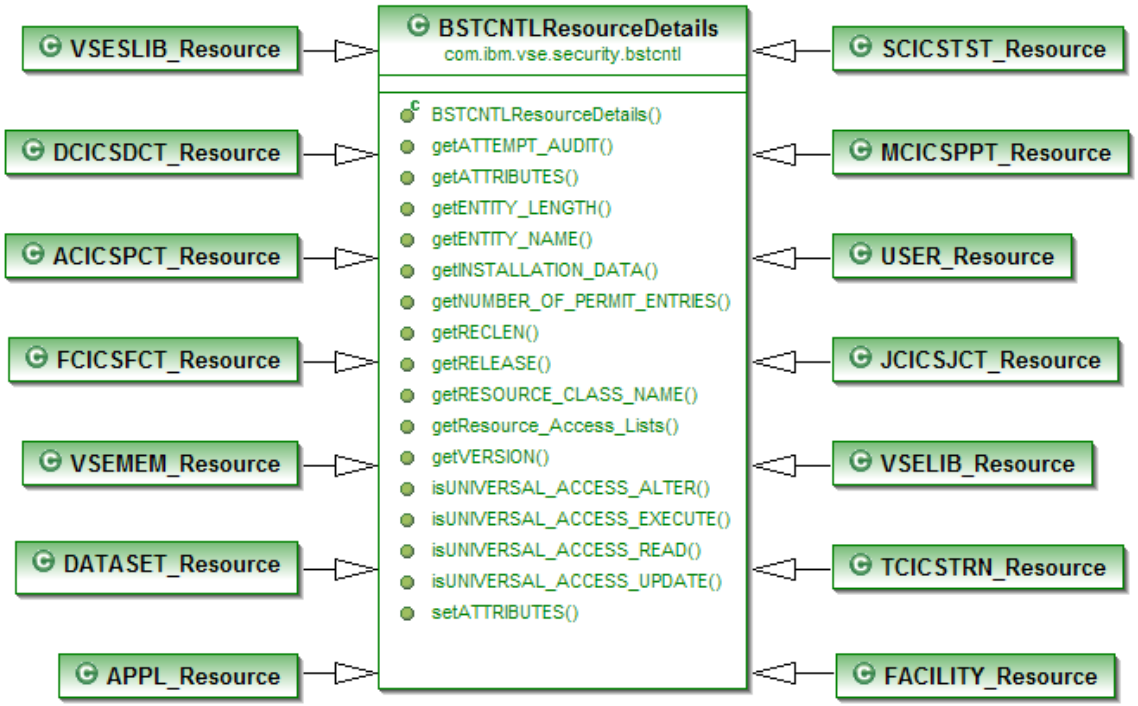

Abbildung 8.5: *com.ibm.vse.security.resource* Package

Klasse DTSECTAB

Die Klasse `DTSECTAB` sammelt und wertet den Inhalt des aktiven DTSECTAB-Members aus und stellt die Daten als Listen der Typen `DTSECTABDetails` und `DTSECTABUsers` bereit. Hier stehen drei Konstruktoren zur Verfügung, da die DTSECTAB an mehreren Orten im VSE liegen kann (siehe Abschnitt 8.1.3 auf Seite 71).

Klasse DTSECTABDetails

`DTSECTABDetails` repräsentiert die Berechtigungseinträge auf Libraries, Sublibraries, Members und Files im DTSEBTAB-Member

Klasse DTSECTABUsers

Diese Klasse hält die zwei Systembenutzer `DUMMY` und `FORSEC` und stellt Methoden zur Verfügung um deren Details anzeigen zu können.

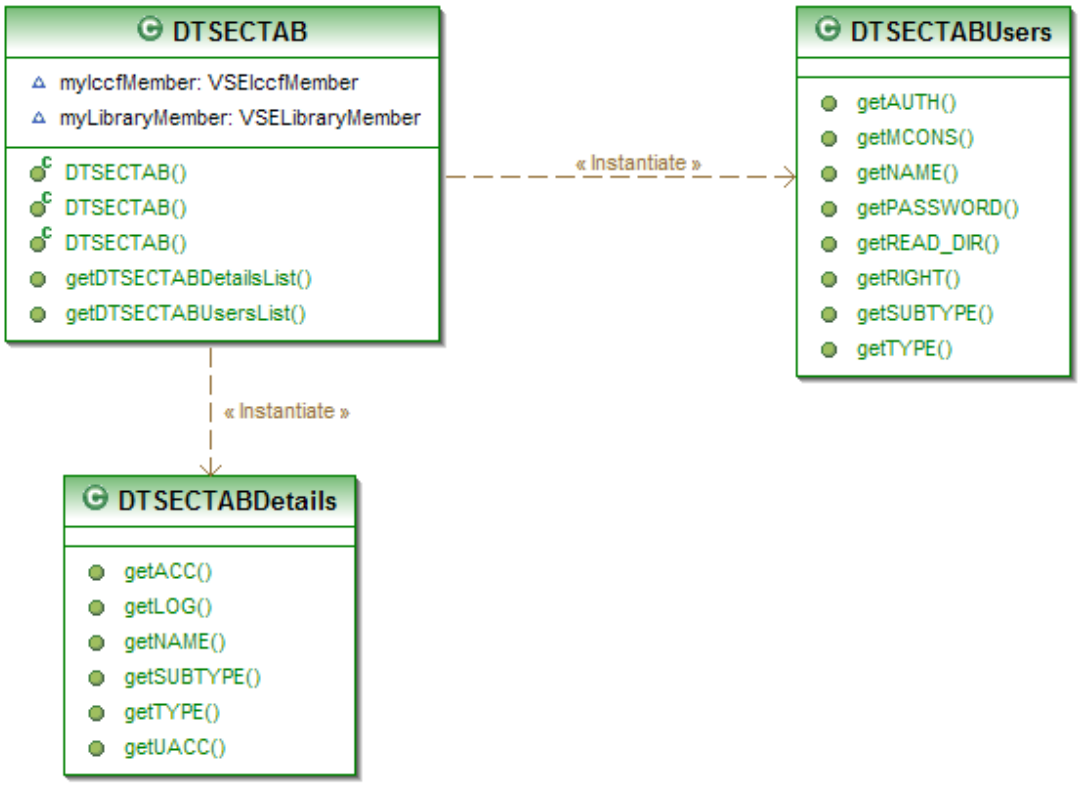

Abbildung 8.6: *com.ibm.vse.security.dtsectab* Package

8.2.6 TCP/IP Package

Mit Hilfe der Klassen (siehe Abbildung 8.7 auf der nächsten Seite) in diesem Package werden die TCP/IP Konfigurationseinstellungen ausgelesen.

Klasse TCP_IP_Config

Die TCP/IP Einstellungen werden mit Hilfe der Klasse `TCP_IP_Config` aus dem Member `IPINIT00.L` (siehe Anhang E auf Seite 121) gelesen und ausgewertet.

Klasse TCP_IP_Config_Details

Diese Klasse ist eine Repräsentation des TCP/IP Members `IPINIT00.L`. Die Parameter sind über getter-Methoden zu erreichen.

8.2.7 Connector Server Package

Diese Klassen (siehe Abbildung 8.8 auf Seite 85) lesen die Connector Server Einstellungen aus.

Klasse ConnectorServerConfig

`ConnectorServerConfig` greift auf die allgemeine Konfigurationsdatei IESVCSRV.Z (siehe Anhang D.3 auf Seite 116) und speichert die geparsten Daten in der Klasse `ConnectorServerConfigDetails`.

Klasse ConnectorServerConfigDetails

`ConnectorServerConfigDetails` repräsentiert das Member IESVCSRV.Z (siehe Anhang D.3 auf Seite 116).

Diese Klassen (siehe Abbildung 8.9 auf Seite 85) lesen die Connector Server Benutzer Einstellungen aus.

Abbildung 8.7: *com.ibm.vse.security.ip* Package

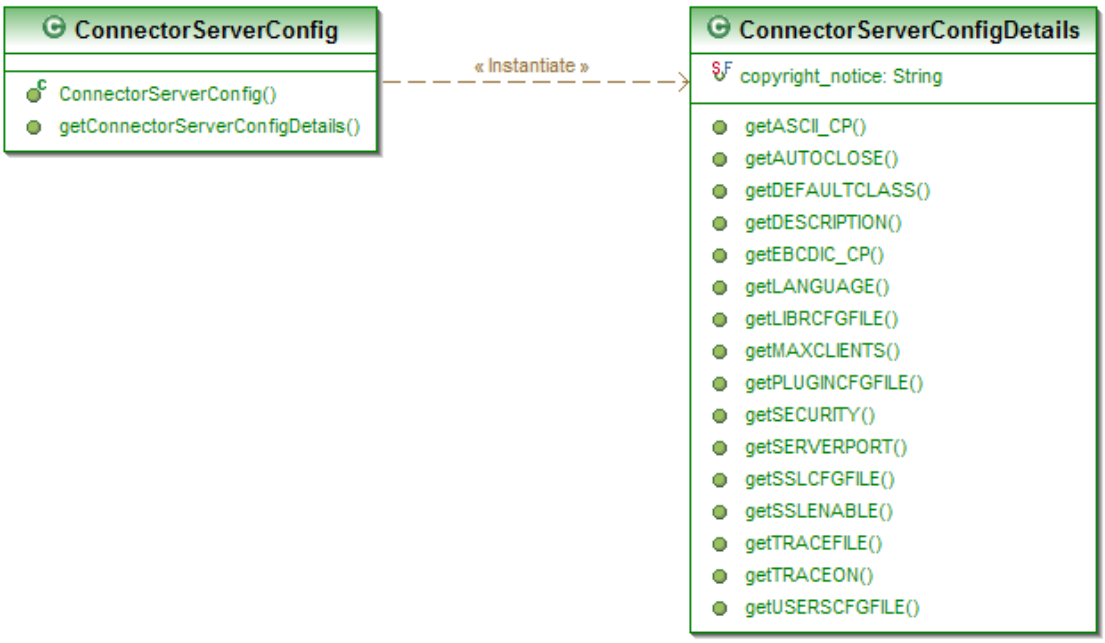

Abbildung 8.8: *com.ibm.vse.security.vcs* Package-Server Config

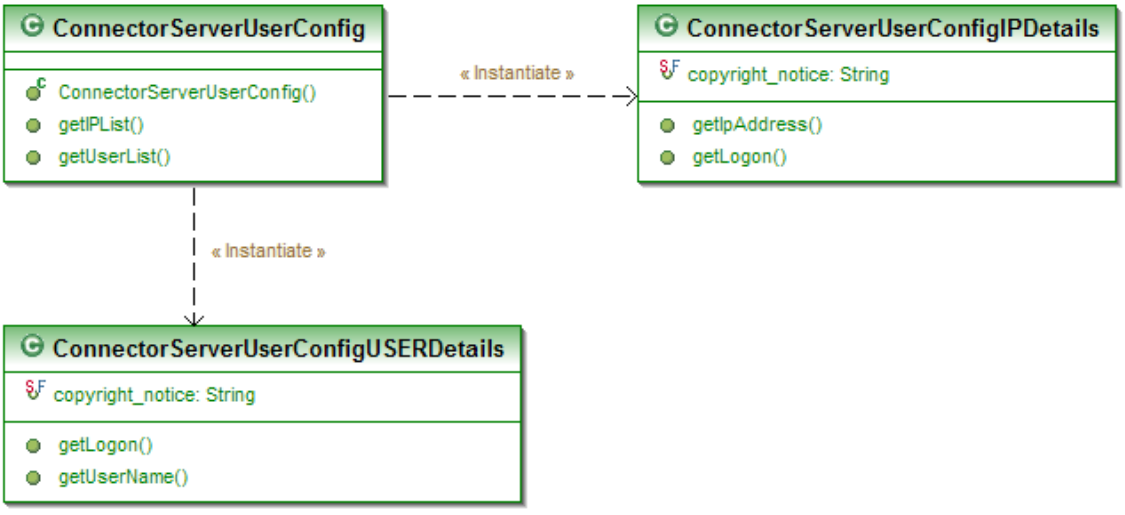

Abbildung 8.9: *com.ibm.vse.security.vcs* Package-Server User Config

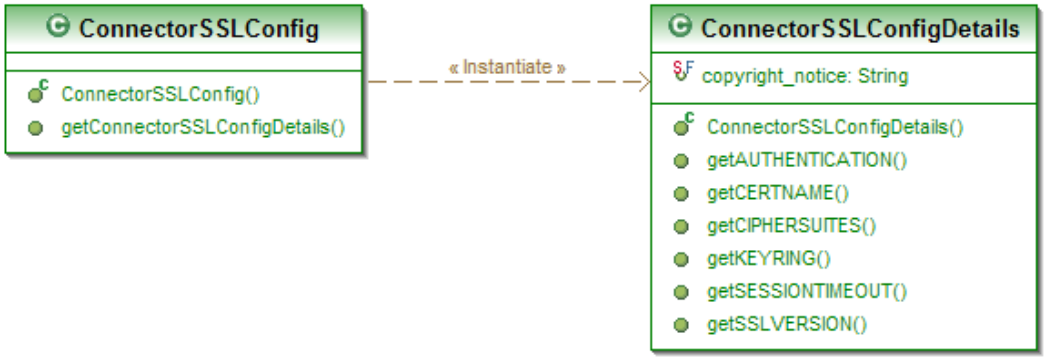

Abbildung 8.10: *com.ibm.vse.security.vcs* Package-SSL Config

Klasse ConnectorServerUserConfig

Die Benutzer-Konfigurationsdatei des Connector Servers wird mit dieser Klasse angesprochen und ausgewertet. Die Details zu den Benutzer- und IP-Adressenberechtigungen werden in den Klassen `ConnectorServerUserConfigIPDetails` und `ConnectorServerUserConfigUSERDetails` gehalten.

Klasse ConnectorServerUserConfigIPDetails

Diese Klasse repräsentiert einen Benutzereintrag in der Connector Server Konfigurationsdatei IESUSERS.Z (siehe Anhang D.1 auf Seite 115)

Klasse ConnectorServerUserConfigUSERDetails

Diese Klasse repräsentiert einen IP-Adresseneintrag in der Connector Server Konfigurationsdatei IESUSERS.Z (siehe Anhang D.1 auf Seite 115)

Diese Klassen (siehe Abbildung 8.10) lesen die Connector SSL Einstellungen aus.

Klasse ConnectorSSLConfig

Diese Klasse liest die SSL Konfigurationsdatei des Connector Servers IESSSLCF.Z (siehe Anhang D.4 auf Seite 117) aus.

Abbildung 8.11: *com.ibm.vse.security.vcs* Package-Librarian Config

Klasse ConnectorSSLConfigDetails

Die SSL-Parameter des Members IESSSLCF.Z (siehe Anhang D.4 auf Seite 117) werden von der Klasse `ConnectorSSLConfigDetails` repräsentiert.

Diese Klasse (siehe Abbildung 8.11) liest die Connector Librarian Einstellungen aus.

Klasse ConnectorLibrarianConfig

Die `ConnectorLibrarianConfig` extrahiert die angegebenen Bibliotheken in der Connector Server Konfigurationsdatei IESLIBDF.Z (siehe Anhang D.2 auf Seite 115) und liefert eine Liste aller Bibliotheken zurück.

8.2.8 Systemdaten Package

Die Klassen (siehe Abbildung 8.12 auf der nächsten Seite) in diesem Package werten den Output des Systemkommandos `SIR` (siehe Anhang B auf Seite 111) aus.

Klasse VSESecurityCommand

Diese Klasse bietet die Funktionalität Kommandos an das VSE System abzuschicken. Als Rückgabewert (Output) wird der Konsoleninhalt geliefert, der dann weiterverarbeitet werden kann. Es können drei verschiedene Arten von Kommandos abgesetzt werden, die jeweils von einem Konstruktor verarbeitet werden.

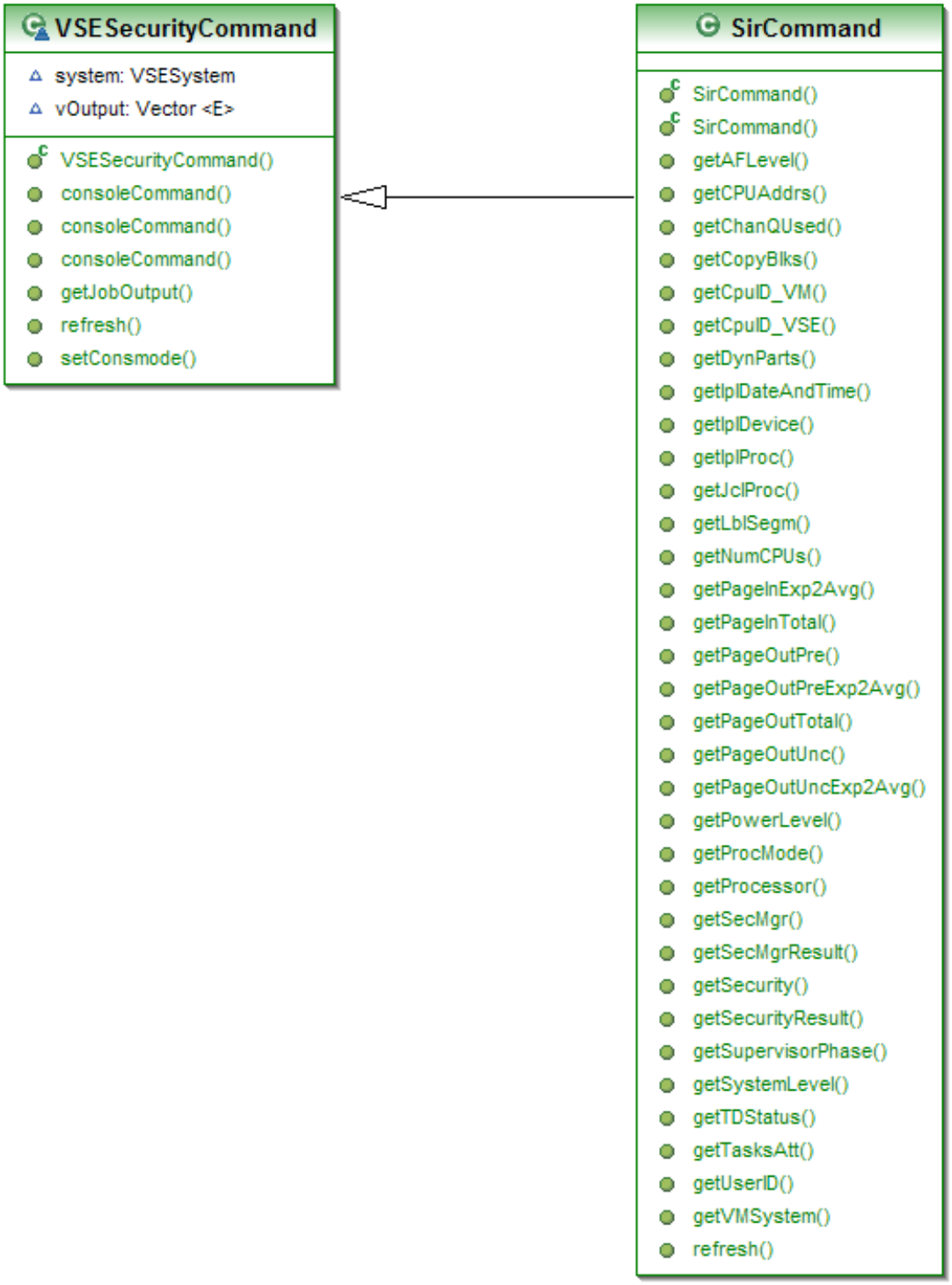

Abbildung 8.12: *com.ibm.vse.security.utils* Package

Klasse SirCommand

Die Klasse Subklasse `SirCommand` erbt von der Klasse `VSESecurityCommand.` Der Output des `SIR` Kommandos wird hier geparst, ausgewertet und durch getter-Methoden zur Verfügung gestellt.

9 Realisierung

Die Abschnitte in diesem Kapitel beschreiben die Implementierung (siehe 9.1) des im Kapitel 8 beschriebenen Designs der VSESecurity Java Klassenbibliothek.

9.1 Implementierung

Im Folgenden wird auf die Umsetzung der Datenerfassung eingegangen. Auf Grund des vertraulichen Aufbaus des BSM.CONTROL.FILE [BCF] und VSE.CONTROL.FILE [VCF] werden in diesem Abschnitt nur exemplarisch die Erfassung der Daten aus Konfigurationsdateien und der DTSECTAB behandelt.

Die Daten aus BCF und VCF werden wie in der Designphase (siehe Abschnitt 8.1.1 auf Seite 68 und 8.1.2 auf Seite 69) beschlossen, über eine Bit-Schablone (confidential) ausgelesen. Diese Schablone wird über jeden VSAM-Datensatz (siehe Abschnitt 4.4 auf Seite 37) gelegt. Nach diesem Prozess werden die extrahierten Inhalte den Klassen `BSTCNTLDetails` oder `Control_File_User_Details` zugewiesen.

9.1.1 Datenerfassung

Die Erfassung der Daten aus den VSE Konfigurationsdateien und dem DTSECTAB Member erfolgt nach demselben Prinzip. Nachdem die Dateien (oder Members) vom VSE System in ein Java-File-Object geladen worden sind, kann der Inhalt zeilenweise ausgelesen werden. Die Herausforderung bestand hier nun in der Konzeption und Umsetzung eines schnellen und möglichst allgemein gehaltenen Parser, der die Parameter aus den Strings extrahiert. Dabei war zu beachten, dass der Parser bei Veränderungen der Konfigurationsdateien in kommenden VSE-Versionen und bei leicht fehlerhafter Konfiguration seitens des Administrators weiterhin funktioniert. Der folgende *code snippet* zeigt einen Teil des DTSECTAB Parsers.

```
if (msg2.indexOf("NAME") != -1) {
  col1 = msg2.indexOf("NAME");
  col2 = msg2.indexOf("=", col1);
  if (msg2.indexOf("EJECT") != -1) {
    col3 = msg2.indexOf("EJECT");
  } else
    col3 = msg2.length();
  myDTSECTABObject.NAME = (msg2.substring(col2 + 1, col3));
}
```

In diesem kleinen Beispiel wird geprüft ob in einer Zeile der DTSECTAB (siehe 8.1.3 auf Seite 71) der String `NAME` vorkommt. Wenn dies der Fall ist ermittelt der Parser an dieser Stelle den Parameter `NAME` und weißt dem `DTSECTABDetails Object` (siehe 8.2.5 auf Seite 80) diesen Wert zu (`myDTSECTABObject.NAME = (msg2.substring(col2 + 1, col3)`).

Für jede Konfigurationsdatei wurde in dieser Arbeit einen auf diese Datei angepassten Parser geschrieben die alle wichtigen Sicherheitsparameter aus den VSE Dateien auslesen.

9.1.2 Datenzusammenfassung

Nachdem alle Daten aus dem VSE ausgelesen wurden, müssen diese noch zusammengefasst werden, so dass zum Beispiel alle Sicherheitsdaten eines Benutzers aus dem VCF, dem DTSECTAB Member und den Konfigurationsdateien über die Benutzerklasse `UserDetails` abrufbar sind. Wie schon in der `PrepareMergedData` Klassenbeschreibung (siehe Abschnitt 8.2.2 auf Seite 76) erläutert, übernimmt diese Aufgabe die Klasse `PrepareMergedData`. Sie wird von Methoden der Klasse `VSESecurity` aufgerufen, die die gesammelten Daten zum Abgleich an Methoden der `PrepareMergedData` Klasse übergibt.

```
/**
* returns a list of user access entries in the DTSECTAB
*/
public ArrayList getDTSECTAB_User_Access(String userId){
  ArrayList accessDTSECTAB = new ArrayList();
  accessDTSECTAB = mPrepareMergedData. getDT-
SECTAB_User_Access(mControlFile.
  getUsers(userId), getDTSECTABDetailsList());

return accessDTSECTAB;
}
```

Die Methode `getDTSECTAB_User_Access` prüft mit Hilfe der Benutzerdaten (`mControlFile.getUsers(userId)`) und einer Liste aller DTSECTAB-Einträgen (`getDTSECTABDetailsList()`) auf welche Libraries, Sublibraries, Members und Files ein Benutzer Zugriff hat. Die `PrepareMergedData` Methode `getDTSECTAB_User_Access` vergleicht nun die Einträge des Benutzers in seinen Zugriffsberechtigungsklassen mit denen in der DTSECTAB. Alle Übereinstimmungen (Zugriffsberechtigungen) werden in einer Liste zurückgegeben.

Im Folgenden wird die Methode `getDTSECTAB_User_Access` in mehreren Abschnitten erläutert.

```
/**
* returns all DTSECTAB entries which the user is accessed
* @param user
* @param details_List_DTSECTAB
* @return
*/
protected ArrayList getDTSECTAB_User_Access(UserDetails user,
 ArrayList details_List_DTSECTAB)
{
  ArrayList accessList = new ArrayList();
  String accessListEntries = "";
  //CHECKS FOR EVERY DTSECTAB ENTRY
  for (int i = 0; i < details_List_DTSECTAB.size(); i++) {
    //THE DETAILS OF A DTSECTAB ENTRY
    DTSECTABDetails myDTSECTAB_Details =
(DTSECTABDetails) details_List_DTSECTAB.get(i);
    ArrayList classResourceList = new ArrayList();
```

Hier werden alle DTSECTAB Einträge und das Benutzerobjekt an die Methode übergeben. Danach wird jeder einzelne DTSECTAB Eintrag geprüft.

```
        //CHECKS IF UNIVERSAL ACCESS OR SPECIAL USER ACCESS
        if (myDTSECTAB_Details.getACC() != null) {
          //GETS SPECIAL USER ACCESS ENTRY (E.G. ACC=(4,5-9))
          String firstIntClass =     myDT-
    SECTAB_Details.getACC().substring(0, 1);
          String lastIntClass = myDTSECTAB_Details.getACC().substring(
          myDTSECTAB_Details.getACC().length()-1,
          myDTSECTAB_Details.getACC().length());
          //SPLITS THE ENTRY TO SEPARATE THE CLASSES
          String[] classArray = myDTSECTAB_Details.getACC().split(",");
          for (int a = 0; a < classArray.length; a++) {
```

Zuerst wird geprüft ob bei einem Eintrag `UNIVERSAL ACCESS (UACC)` vorliegt und der Benutzer deshalb Zugriffsrechte hat. Wenn nicht wird der `ACC` Parameter ausgelesen. Die Berechtigungsangaben im ACC Parameter sind durch Kommas getrennt und werden danach getrennt.

```
    //CKECKS IF THERE IS A RANGE
            if (classArray[a].indexOf("-") != -1) {
              firstIntClass = classArray[a].substring(0, 1);
              lastIntClass = classArray[a].substring(
              classArray[a].length()-1, classArray[a].length());
              for (int ra = Integer.parseInt(firstIntClass); ra <= Inte-
    ger.parseInt(lastIntClass); ra++) {
                String raa = String.valueOf(ra);
                classResourceList.add(raa);
              }
            } else
              classResourceList.add(classArray[a]);
            }
```

In diesem Methodenabschnitt wird unterschieden ob die Zugriffsberechtigungsklassen einzeln oder durch eine von bis Angabe angegeben sind. Je nach Angabe werden die Infor-

mationen ausgeparst.

```
//CHECKS IF THE USER CLASS AND DTSECTAB CLASS ENTRIES MATCHES
byte[] myDTSECTAB_Access = user.getControl_File_User()
 .getDtsectab_Classes_Data();
for (int ii = 0; ii < myDTSECTAB_Access.length; ii++) {
  String right = "";
  if (myDTSECTAB_Access[ii] == 1) {
    right = "CONNECT";
  } else if (myDTSECTAB_Access[ii] == 2) {
    right = "READ";
  } else if (myDTSECTAB_Access[ii] == 3) {
    right = "UPDATE";
  } else if (myDTSECTAB_Access[ii] == 4) {
    right = "ALTER";
}
```

Erst jetzt werden die Informationen aus einem einzelnen DTSECTAB Eintrag mit den Angaben im Benutzerprofil verglichen. Wenn eine Übereinstimmung der Zugriffsberechtigungsklassen des Benutzers und eines DTSECTAB Eintrags vorkommt, ist noch zu prüfen welche Berechtigung der Benutzer auf hat. Je nach Angabe im Benutzerprofil in der Zugriffsberechtigungsklasse hat der Benutzer das Zugriffsrecht CONNECT, READ, UPDATE oder ALTER.

```
            for (int j = 0; j < classResourceList.size(); j++) {
              int classResourceEntry = Integer
              .parseInt(((String) classResourceList.get(j)));

              //CHECKS THE CLASS ENTRIES AND FILLS THE ACCESSLIST
              if (ii + 1 == classResourceEntry
                && myDTSECTAB_Access[ii] != 0) {
                accessListEntries = "TYPE: "
                + myDTSECTAB_Details.getTYPE() + ", NAME: "
                + myDTSECTAB_Details.getNAME() + ", ACCESS: "+ right;
                accessList.add(accessListEntries);
              }
            }
          }
        }
      }
      return accessList;
      }
```

Im letzten Abschnitt der Methode wird die Zugriffsliste des Benutzers mit den DTSECTAB Einträgen und die dazugehörigen Berechtigungen gefüllt und an das Benutzerobjekt zurückgeliefert.

9.2 Code Statistik

Diese Statistik gibt einen Einblick über die Komplexität und den Aufwand der Implementierung der Java Klassenbibliothek VSESecurity. Diese Statistik wurde mit einer internen IBM Anwendung erstellt. Kommentare und Javadoc Anweisung werden nicht berücksichtigt. Des Weiteren zählen Statements über zwei Zeilen natürlich nur einmal.

Tabelle 9.1: Code Statistik der VSESecurity Klassenbibliothek

Package Name	Anzahl der Klassen	Anzahl der Statements
com.ibm.vse.security	4	733
com.ibm.vse.security.bstcntl	5	834
com.ibm.vse.security.dtsectab	3	309
com.ibm.vse.security.iescntl	3	513
com.ibm.vse.security.ip	2	227
com.ibm.vse.security.resource	14	44
com.ibm.vse.security.vcs	8	339

9.3 Verwendung des Java-based Connectors

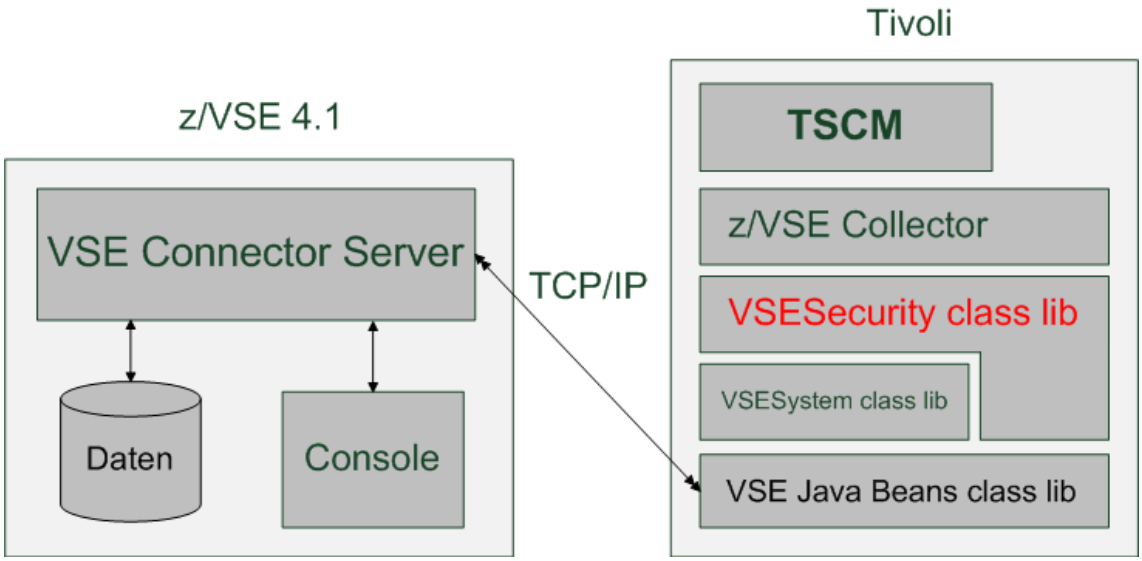

Abbildung 9.1: VSE Java-based connector: Zugriff auf sicherheits-relevante VSE Informationen

Über den Java-based Connector (siehe Abschnitt 4.3 auf Seite 35) greift die Klassenbibliothek auf VSE Parameter zu. Hierbei spielt die Klassenbibliothek VSESystem[1] eine wichtige Rolle, da mit ihrer Hilfe ein VSE System abgebildet werden kann. Auf dieser VSE-Abbildung setzt die VSESecurity Klassenbibliothek auf. Der Tivoli TSCM (siehe

[1] weitere Informationen siehe http://www.ibm.com/servers/eserver/zseries/zvse/products/connectors.html#vsesystem

Abschnitt 3.3.2.1 auf Seite 24) verwendet die VSESecurity Bibliothek um ein *State Monitoring* über die ausgelesenen VSE Sicherheitsparameter bereitzustellen.

9.4 Test

Um die entwickelte Bibliothek VSESecurity an einem VSE System testen zu können, wurde ein z/VSE 4.1 Testsystem aufgesetzt. In diesem wurden mehrere Benutzer mit verschiedenen Benutzerrollen und Berechtigungen angelegt. Weiter wurden DTSECTAB Einträge, BSTCNTL (BSM.CONTROL.FILE) Gruppen- und Benutzereinträge mit verschiedenen Zugriffsrechten hinzugefügt, so dass das Testsystem möglichst realitätsnah konfiguriert war. Diese Einstellungen im System wurden mit Hilfe von BSTADMIN Kommandos 5.2.1 auf Seite 48 umgesetzt, die jeweils in einem Job zusammengefasst wurden. Diese Jobs sind zum Nachschlagen im Anhang F.2 auf Seite 125 aufgeführt.

Für den Test wurde die Testklasse `SecurityStart` mit Methoden ausgestattet, die alle Klassen, wie sie im Abschnitt 8.2 auf Seite 73 aufgeführt sind, testet. Ein Beispiel-Output von der Methode `testDTSECTAB_User_Access()`, welche prüft auf welche Libraries, Sublibraries und Files ein Benutzer welchen Zugriff hat, wird im Folgenden gezeigt.

Beispiel-Output der Testmethode: `testDTSECTAB_User_Access()`

```
TYPE: SUBLIB, NAME: IJSYSRS.SYSLIB, ACCESS: READ
TYPE: SUBLIB, NAME: IJSYSRS.SYSLIB, ACCESS: CONNECT
TYPE: SUBLIB, NAME: IJSYSRS.SYSLIB, ACCESS: UPDATE
TYPE: MEMBER, NAME: IJSYSRS.SYSLIB.CPUVAR*, ACCESS: READ
TYPE: MEMBER, NAME: IJSYSRS.SYSLIB.CPUVAR*, ACCESS: ALTER
```

Diesem Output ist zu entnehmen, dass der abgefragte Benutzer die hinten aufgeführten Zugriffsrechte (`READ, CONNECT, UPDATE oder ALTER`) auf die jeweiligen Bibliotheken oder Members hat. Alle weiteren Testmethoden und deren Funktionalität sind der JavaDoc zu entnehmen.

Diese ausgegebenen Zugriffsrechte dieses Benutzers werden dann beispielsweise in einem *State Monitoring* des TSCM ausgewertet.

10 Ausblick und Zusammenfassung

10.1 Ausblick

Die entwickelte Java Klassenbibliothek VSESecurity kann verwendet werden, um alle sicher-heits-relevanten Informationen eines VSE Betriebssystems auszulesen und zu überwachen. Speziell für VSE Kunden, die eine große Anzahl an Benutzern und Ressourcen verwalten und administrieren müssen, stellt dieser Ansatz in Verbindung mit dem TSCM (siehe Abschnitt 3.3.2.1 auf Seite 24) eine effiziente, kostengünstige und zentrale Lösung zur Überwachung der VSE Systemsicherheit dar.

Weiter kann die Klassenbibliothek als Basis für zukünftige Anwendungen, die Zugriff auf sicherheits-relevante VSE Parameter benötigen, dienen. In diesem Zusammenhang könnte beispielsweise der 'VSE Navigator' (siehe 7.1 auf Seite 61) die VSESecurity Bibliothek verwenden, um die Sicherheitseinstellungen des ganzen Systems auszulesen und über eine GUI zu visualisieren. Außerdem gewährleistet die Skalierbarkeit der Klassenbibliothek eine beliebige Abdeckung nachfolgender Anforderungen.

Die in dieser Arbeit implementierte Klassenbibliothek soll äquivalent zu anderen z/VSE Connector Komponenten auf der VSE Produktseite[1] als Download bereit gestellt werden.

10.2 Zusammenfassung

Zu Beginn der Diplomarbeit wurden die gestellten Anforderungen an die zu entwickelnde Klassenbibliothek analysiert und erweitert. Die aus dem System auszulesenden und erforderlichen VSE Sicherheitsparameter wurden mit Hilfe dieser Anforderungen ermittelt. Nach der Lokalisierung dieser sicherheits-relevanten Daten im VSE wurde das dieser Klassenbibliothek zu Grunde liegende Datenmodell entworfen. Im Folgenden wurden die verschiedenen Zugriffsmethoden auf VSE Daten diskutiert und die für diese Zwecke sinnvollsten Zugriffsarten gewählt.

[1] http://www.ibm.com/servers/eserver/zseries/zvse/products/connectors.html

Mit den Ergebnissen der Analyse- und Designphase wurde eine Java Klassenbibliothek implementiert, die eine umfassende Abbildung sicherheits-relevanter VSE Daten strukturiert und objektorientiert anbietet und damit eine zentrale, automatisierte Überwachung der VSE Sicherheitsparameter ermöglicht.

Persönlich bekam ich tiefe Einblicke in das IBM Betriebssystem z/VSE und dessen Sicherheitskonzept und Architektur. Diese Arbeit bot mir viele Gelegenheiten, meine im Studium erlangten Fähigkeiten und Erfahrungen in den Bereichen Java-Programmierung, Betriebssysteme, Security und Software-Architektur anzuwenden und zu erweitern.

11 Abbildungsverzeichnis

12 Tabellenverzeichnis

Literaturverzeichnis

[KW03] Karl E. Wiegers. Software Requirements. Microsoft Press, 2, Ausgabe, 2003

[ZGK04] Zuser, Grechenig, Köhle: Software Engineering mit UML und dem Unified Process, Pearson Studium, München, 2004

[zOSB07] Introduction to the New Mainframe: Security, Februar 2007, IBM Corp., SG24-6776, verfügbar unter http://www.redbooks.ibm.com/abstracts/sg246776.html?Open

[MAISEC] Introduction to the New Mainframe: z/OS Basics, Juli 2006, IBM Corp., SG24-6366, verfügbar unter http://www.redbooks.ibm.com/abstracts/sg246366.html?Open

[zVSEB06] Introduction to the New Mainframe: z/VSE Basics, Oktober 2006, IBM Corp., SG24-7436, verfügbar unter http://www.redbooks.ibm.com/abstracts/sg247436.html?Open

[zVSEAd07] Administration IBM z/VSE, März 2007, Version 4 Release 1, IBM Corp., SC33-8304, verfügbar unter http://www.ibm.com/servers/eserver/zseries/zvse/documentation/#vse

[zVSECo05] z/VSE e-business Connectors User's Guide, März 2005, Version 3 Release 1, IBM Corp., SC33-8310, verfügbar unter http://www.ibm.com/servers/eserver/zseries/zvse/documentation/#conn

[Tiv99] An Introduction to Tivoli Enterprise, Oktober 1999, IBM Corp.

[TSCM05] IBM Tivoli Security Compliance Manager, August 2005, Version 5 Release 1

[ESMCA99] CA-Top Secret User Guide, April 1999, Version 1 Release 3, IBM Corp., weitere Informationen unter http://www.ca.com/us/products/product.aspx?id=141

[ESMBI00] BIM-ALERT/VSE-Security Administraor's Guide, Dezember 2000, Release 5.1, weitere Informationen unter http://www.e-vse.com/csi-products/BIM-Alert-VSE/BIM-Alert-VSE.htm

[zVSETCP06] TCP/IP for VSE-User Guide, April 2006, Version 1, Release 5E Beta, weitere Informationen unter http://www.ibm.com/servers/eserver/zseries/zvse/documentation/index.html#tcpip

[wwVSECo] Informationen zu den VSE Connectors sind unter http://www.ibm.com/servers/eserver/zseries/zvse/products/connectors.html#conn zu finden

[wwVSE] Allgemeine Informationen zum z/VSE System: http://www.ibm.com/servers/eserver/zseries/zvse/

[wwTiv] Allgemeine Informationen zu den Tivoli Produkten: http://www.ibm.com/software/tivoli/

A DTSECTAB Format

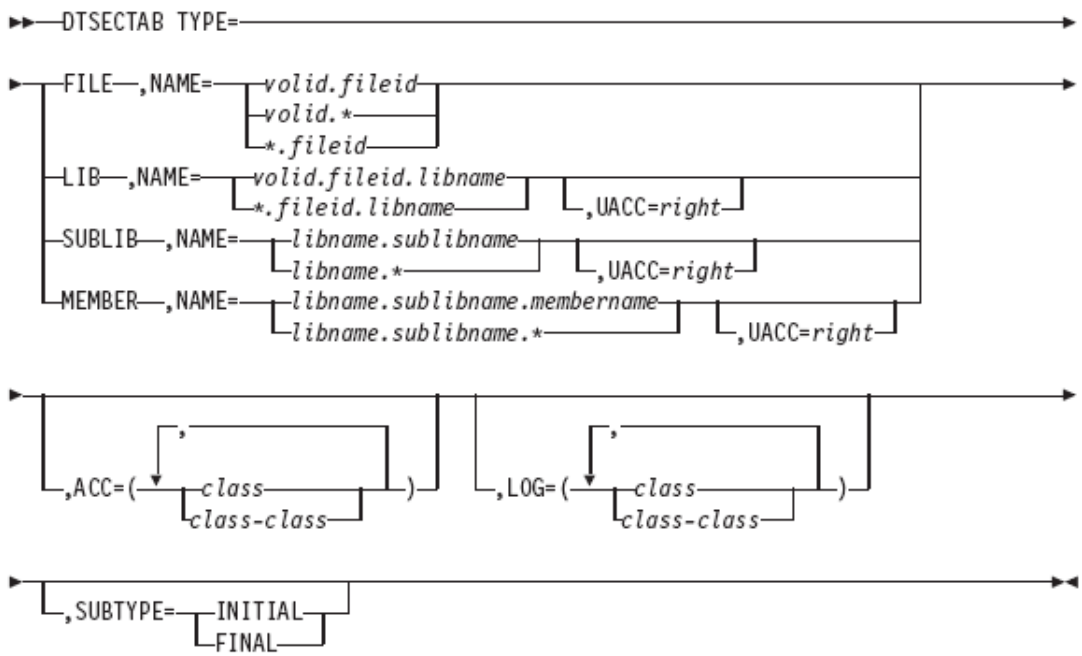

Abbildung A.1: siehe [zVSEAd07] - Format und Aufbau der DTSECTAB

A.1 DTSECTAB Auszug

```
*****************************************************
*                                                   *
*   5686-CF7 (C) COPYRIGHT IBM CORP. 1984, 2004     *
*                                                   *
*****************************************************
TITLE 'DTSECTAB-SECURITY TABLE FOR RESOURCES'
*********************************************************************
PUNCH ' CATALOG DTSECTRC.OBJ  REP=YES'
SPACE 3
*-------------------------------------------------------------------*
```

```
*                                                                     *
*       STATIC PART OF DTSECTAB                                       *
*                                                                     *
*---------------------------------------------------------------------*
*       THIS PART IS SHIPPED AS A-BOOK IN IJSYSRS.SYSLIB.DTSECTRC.    *
*       IF CHANGED, THE USER SHOULD PUT HIS VERSION UNDER THE SAME    *
*       NAME IN PRD2.SAVE, AS IBM SERVICE IS DONE ON THE MEMBERS      *
*       CONTAINED IN IJSYSRS.SYSLIB.                                  *
*       (THE JOB TO BUILD A DTSECTAB LOOKS FIRST IN PRD2.SAVE FOR     *
*       DTSECTRC).                                                    *
*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
*               IBM SUPPLIED USERS                                    *
*---------------------------------------------------------------------*
*** USER DUMMY HAS NO SPECIAL SECURITY RIGHTS.USED TO RESET INHERITANCE
*** IT AVOIDS GETTING TOO MANY RIGHTS WHILE LOADING POWER JOBS DURING
*** ASI.
*** YOU SHOULD NOT DEFINE AN II USER WITH THE NAME 'DUMMY'.
*---------------------------------------------------------------------*
DTSECTAB TYPE=USER,                                             C
NAME=DUMMY,                                           C
PASSWRD=DUMMY,                                        C
AUTH=NO,                                              C
SUBTYPE=INITIAL
SPACE 3
*---------------------------------------------------------------------*
*** USER FORSEC HAS ALL ACCESS RIGHTS. THEREFORE, THE PASSWORD NEEDS
*** TO BE CHANGED AFTER INITIAL INSTALLATION.
*---------------------------------------------------------------------*
DTSECTAB TYPE=USER,                                             C
NAME=FORSEC,                                          C
PASSWRD=FORSEC,                                       C
READDIR=YES,                                          C
MCONS=YES,                                            C
AUTH=YES,                                             C
RIGHT=BTRANS
*---------------------------------------------------------------------*
*               END OF IBM SUPPLIED USERS                             *
*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
*                                                                     *
*       FOLLOWING IS THE Z/VSE 3.1 SUPPLIED PART OF THE DTSECTAB      *
*       THAT DEFINES A MINIMUM SET OF RESOURCES TO BE PROTECTED.      *
*                                                                     *
```

```
*------------------------------------------------------------------*
****** LIBRARIES*********************************************************
******   IJSYSRS
DTSECTAB TYPE=LIB,                                          C
NAME=DOSRES.VSE.SYSRES.LIBRARY.IJSYSRS,                C
UACC=CON
DTSECTAB TYPE=SUBLIB,                                       C
NAME=IJSYSRS.SYSLIB,                                   C
ACC=1-4,7,                                             C
LOG=2-6,9,                                             C
UACC=CON
DTSECTAB TYPE=MEMBER,                                       C
NAME=IJSYSRS.SYSLIB.*,                                 C
UACC=READ
******   CPUVAR* IS USED BY VARIOUS JOBSTREAMS TO SAVE PARAMETERS
DTSECTAB TYPE=MEMBER,                                       C
NAME=IJSYSRS.SYSLIB.CPUVAR*,                           C
ACC=9,2-3,                                             C
UACC=UPD
******  FILES ***********************************************************
DTSECTAB TYPE=FILE,                                          C
NAME=*.VSE.CONTROL.FILE
DTSECTAB TYPE=FILE,                                          C
NAME=*.VSE.BSTCNTL.FILE
DTSECTAB TYPE=FILE,                                          C
NAME=DOSRES.VSE.POWER.QUEUE.FILE
DTSECTAB TYPE=FILE,                                          C
NAME=SYSWK1.VSE.POWER.DATA.FILE,                       C
SUBTYPE=FINAL
EJECT
*------------------------------------------------------------------*
*                 END OF Z/VSE DTSECTAB                            *
*------------------------------------------------------------------*
SPACE 3
END
```

B SIR-Command Output

Output des SIR Kommandos

```
AR 0015 CPUID  VM = 0025981020640000      VSE = FF00000720640000
AR 0015 VM-SYSTEM = VM/ESA  (LPAR)  4.2.0       0000
AR 0015 PROCESSOR = 2064-00         USERID    = VSE01
AR 0015 PROC-MODE = ESA (64-BIT)    IPL(150)    15:32:06      05/07/2003
AR 0015 SYSTEM    = VSE/ESA         2.7.0 GA                  01/14/2003
AR 0015             VSE/AF          6.7.0       GA-LEVEL      12/02/2002
AR 0015             VSE/POWER       6.7.0       DY BASE       12/02/2002
AR 0015 IPL-PROC  = $IPLESA         JCL-PROC  = $$JCL
AR 0015 SUPVR     = $$A$SUPX        TURBO-DISPATCHER (40) ACTIVE
AR 0015                             HARDWARE COMPRESSION ENABLED
AR 0015 SEC. MGR. = BASIC           SECURITY  = ONLINE
AR 0015 VIRTCPU   = 0000:01:18.577         CP = 0000:00:07.717
AR 0015 CPU-ADDR. = 0000(IPL)   ACTIVE
AR 0015   ACTIVE  = 0000:00:17.880   WAIT = 0023:07:47.848
AR 0015   PARALLEL= 0000:00:05.440   SPIN = 0000:00:00.000
AR 0015 CPU-ADDR. = 0001        ACTIVE
AR 0015   ACTIVE  = 0000:00:13.563   WAIT = 0023:07:51.570
AR 0015   PARALLEL= 0000:00:04.640   SPIN = 0000:00:00.000
AR 0015 CPU-ADDR. = 0002        ACTIVE
AR 0015   ACTIVE  = 0000:00:10.326   WAIT = 0023:07:56.439
AR 0015   PARALLEL= 0000:00:03.357   SPIN = 0000:00:00.000
AR 0015 CPU timings MEASUREMENT INTERVAL    0023:08:16.998
AR 0015 TASKS ATT.= 00024           HIGH-MARK = 00024    MAX = 00164
AR 0015 DYN.PARTS = 00001           HIGH-MARK = 00001    MAX = 00048
AR 0015
AR 0015 COPY-BLKS = 00006           HIGH-MARK = 00030    MAX = 01500
AR 0015 CHANQ USED= 00006           HIGH-MARK = 00011    MAX = 00100
AR 0015 LBL.-SEGM.= 00008           HIGH-MARK = 00008    MAX = 00717
AR 0015 PGIN  TOT.= 0000000202      EXP.AVRGE.= 0000000001/SEC
AR 0015 PGOUT TOT.= 0000000446
AR 0015       UNC.= 0000000297      EXP.AVRGE.= 0000000000/SEC
AR 0015       PRE = 0000000149      EXP.AVRGE.= 0000000000/SEC
AR 0015 LOCKS EXT.= 0000002163      LOCKS INT.= 0000013901
AR 0015      FAIL = 0000000022           FAIL = 0000000044
AR 0015 LOCK I/O  = 0000000000      LOCK WRITE= 0000000000
AR 0015 1I40I  READY
```

C VSESecurity Konfigurationsdatei

Beispiel: vsesecurity.properties Konfigurationsdatei

```
# This is the properties file for the VSESecurity Library
#
# You have to add all necessary parameters!
#
# Prompt for IP address, user ID and password.
# xxx.xxx.xxx.xxx
ipAddr   = 1.22.333.4
port     = 3333
userID   = ANYUSER
password = PWD
################################################
## ATTENTION ##
## Define only one of these three possibilities
# 1-VSE system data for DTSECTAB file
iccflibrary = 59
iccfname    = DTSECTRC
# 2-VSE system data for DTSECTAB file in VSE Library
vselibrary    =
vsesublibrary =
vsename       =
vsetype       =
# 3-VSE system data for DTSECTAB file in local file
filename =
################################################
# VSE system data for Connector Librarian Config File
ConnectorLibrarianConfig_Library    = PRD2
ConnectorLibrarianConfig_Sublibrary = CONFIG
ConnectorLibrarianConfig_Name       = IESLIBDF
ConnectorLibrarianConfig_Type       = Z
# VSE system data for Connector Server Config File
ConnectorServerConfig_Library    = PRD2
ConnectorServerConfig_Sublibrary = CONFIG
ConnectorServerConfig_Name       = IESVCSR2
ConnectorServerConfig_Type       = Z
# VSE system data for Connector Server User Config File
ConnectorServerUserConfig_Library    = PRD2
ConnectorServerUserConfig_Sublibrary = CONFIG
```

```
ConnectorServerUserConfig_Name       = IESUSERS
ConnectorServerUserConfig_Type       = Z
# VSE system data for Connector SLL Config File
ConnectorSSLConfig_Library    = PRD2
ConnectorSSLConfig_Sublibrary = CONFIG
ConnectorSSLConfig_Name       = IESSSLCF
ConnectorSSLConfig_Type       = Z
# VSE system data for TCP/IP
TCPIP_Library    = PRD2
TCPIP_Sublibrary = CONFIG
TCPIP_Name       = IPINIT00
TCPIP_Type       = L
```

D VSE Connector Server

Sicherheits-Konfigurationsdateien des VSE Connector Server

D.1 IESUSERS.Z

VSE Connector Server Konfigurationsdatei für Benutzer

```
* ****************************************************************
* VSE CONNECTOR SERVER USER SECURITY CONFIGURATION MEMBER
* YOU CAN EITHER ALLOW OR DENIE THE LOGON FOR A SPECIFIED
* USER OR IP OR GROUP OF USERS AND IP ADDRESSES.
* ****************************************************************
* USERS FROM THIS IP'S ARE ALLOWED TO LOGON
* UNCOMMENT THE SAMPLES AND MODIFY THEM
* ****************************************************************
IP   = *,             LOGON = ALLOWED
* IP = 9.164.123.456, LOGON = DENIED
* IP = 9.165.*      , LOGON = DENIED
* IP = 10.0.0.*     , LOGON = ALLOWED
* ****************************************************************
* THIS USERS ARE ALLOWED TO LOGON
* UNCOMMENT THE SAMPLES AND MODIFY THEM
* ****************************************************************
USER = *,             LOGON = ALLOWED
* USER = BOBY,        LOGON = ALLOWED
* USER = SYS*,        LOGON = DENIED
```

D.2 IESLIBDF.Z

VSE Connector Server Konfigurationsdatei für Bibliotheken

```
* ****************************************************************
* LIBRARIAN CONFIGURATION MEMBER FOR VSE CONNECTOR SERVER
* ****************************************************************
*
* ADD THE NAME OF YOUR LIBRARIES TO THIS MEMBER F YOU WANT TO
* HAVE ACCESS TO THEM WITH THE VSE CONNECTOR SERVER.
```

```
*
* NOTE:  EACH LINE SPECIFIES ONLY ONE LIBRARY
*
* ********************************************************************
PRD1
PRD2
PRIMARY
IJSYSRS
CRYPTO
SYSDUMP
```

D.3 IESVCSRV.Z

VSE Connector Server Konfigurationsdatei

```
; ********************************************************************
;      MAIN CONFIGURATION MEMBER FOR VSE CONNECTOR SERVER
; ********************************************************************
; ********************************************************************
; TRACING SPECIFIC SETTINGS:
;-TRACEON    : A 32 BIT HEX VALUE PREFIXED WITH '0X'
;                0X00000000 IS OFF, 0XFFFFFFFF IN ON
;-TRACEFILE  : DESTINATION FOR TRACE MESSAGES
;                DD:SYSLOG, DD:SYSLST OR DD:LIB.SLIB(NAME.TYPE)
; ********************************************************************
TRACEON      = 0X00000000  ; TRACE IS OFF
TRACEFILE    = DD:SYSLOG   ; TRACE GOES TO SYSLOG
; ********************************************************************
; TCP/IP-SERVER SPECIFIC CONFIGURATIONS
;-SERVERPORT : THE TCP PORT WHERE THE SERVER IS LISTENING
;-MAXCLIENTS : THE MAXIMUM NUMBER OF CONCURRENT CLIENTS
;-SSLENABLE  : YES/NO-USE SECURE SOCKET LAYER
; ********************************************************************
SERVERPORT   = 2893
MAXCLIENTS   = 256
SSLENABLE    = NO
; ********************************************************************
; SECURITY CONFIGURATION
;-SECURITY: FULL    -LOGON, RESOURCE AND USER TYPE CHECKING
;             RESOURCE-LOGON AND RESOURCE, BUT NO USER TYPE
;                        CHECKING.
;             LOGON   -LOGON, BUT NO RESOURCE AND USER TYPE
;                        CHECKING
;             NO      -NO LOGON, RESOURCE AND USER TYPE CHECKING
; ********************************************************************
SECURITY = FULL
; ********************************************************************
; TIMEOUT FOR VSAM AUTO CLOSE
;-AUTOCLOSE : THE TIMEOUT IN MSEC FOR VSAM AUTOCLOSE
; ********************************************************************
AUTOCLOSE    = 600
```

```
; ****************************************************************
; DEFAULT CLASS FOR JOBS
;-DEFAULTCLASS : THE JOB CLASS WHERE UTILITY JOBS SHOULD RUN
; ****************************************************************
DEFAULTCLASS = P
; ****************************************************************
; CODE PAGE CONVERSIONS
;-ASCII_CP  : ASCII CODE PAGE
;-EBCDIC_CP : EBCDIC CODE PAGE
; ****************************************************************
ASCII_CP     = IBM-850
EBCDIC_CP    = IBM-1047
; ****************************************************************
; SYSTEM LANGUAGE
;-LANGUAGE : THE SYSTEM'S LANGUAGE (E, G, S, J)
; ****************************************************************
LANGUAGE = E
; ****************************************************************
; DESCRIPTION SENT AS IDENTIFY
;-DESCRIPTION : THIS STRING IS SENT AS IDENTIFY TO THE CLIENT
;                 CHANGE '<YOUR SYSTEM>' TO YOUR SYSTEM'S NAME
; ****************************************************************
DESCRIPTION   = VSE CONNECTOR SERVER ON T29LPAR6
; ****************************************************************
; SUB CONFIGURATION MEMBERS NEEDED FOR VSE CONNECTOR SERVER
;-LIBRCFGFILE   : LIBRARY DEFINITION FILE. CONTAINS THE LIBRARIES
;                   THAT ARE VISIBLE FOR THE VSE CONNECTOR SERVER.
;-USERSCFGFILE  : USER/SECURITY CONFIG FILE. DEFINES ADDITIONAL
;                   SECURITY FOR USERS AND IP ADDRESSES.
;-PLUGINCFGFILE : PLUGIN CONFIG FILE. DEFINES THE PLUGINS THAT
;                   ARE LOADED AT SERVER STARTUP.
;-SSLCFGFILE    : SSL CONFIG FILE. DEFINES SSL PARAMETERS
; NOTE: YOU HAVE TO CHANGE THE NAMES AND LOCATIONS OF THESE MEMBERS
;       IN THIS MEMBER IF YOU MOVE THEM TO ANOTHER LIBRARY!
; ****************************************************************
LIBRCFGFILE   = DD:PRD2.CONFIG(IESLIBDF.Z)
USERSCFGFILE  = DD:PRD2.CONFIG(IESUSERS.Z)
PLUGINCFGFILE = DD:PRD2.CONFIG(IESPLGIN.Z)
SSLCFGFILE    = DD:PRD2.CONFIG(IESSSLCF.Z)
```

D.4 IESSSLCF.Z

SSL Konfigurationsdatei für den VSE Connector Server

```
; ****************************************************************
;      SSL CONFIGURATION MEMBER FOR VSE CONNECTOR SERVER
; ****************************************************************
; ****************************************************************
; SSLVERSION   SPECIFIES THE MINIMUM VERSION THAT IS TO BE USED
;              POSSIBLE VALUES ARE:   SSL30 AND TLS31
; KEYRING      SPECIFIES THE SUBLIBRARY WHERE THE KEY FILES ARE
```

```
;              STORED.
; CERTNAME     NAME OF THE CERTIFICATE THAT IS USED BY THE SERVER
; SESSIONTIMEOUT NUMBER OF SECONDS THAT THE SERVER WILL USE TO
;              ALLOW A CLIENT TO RECONNECT WITHOUT PERFORMING A
;              FULL HANDSHAKE. (86440 SEC = 24 HOURS)
; AUTHENTICATION TYPE OF AUTHENTICATION. POSSIBLE VALUES ARE:
;              SERVER-SERVER AUTHENTICATION ONLY
;              CLIENT-SERVER AND CLIENT AUTENTICATION
;              LOGON -SERVER AND CLIENT AUTENTICATION WITH LOGON
;                       THE CLIENT CERTIFICATE IS USED TO LOGON.
; ****************************************************************
SSLVERSION     = SSL30
KEYRING        = CRYPTO.KEYRING
CERTNAME       = RSA2048
SESSIONTIMEOUT = 86440
AUTHENTICATION = SERVER
; ****************************************************************
; CIPHERSUITES SPECIFIES A LIST OF CIPHER SUITES THAT ARE ALLOWED
; ****************************************************************
CIPHERSUITES =   ; COMMA SEPARATED LIST OF NUMERIC VALUES
01,  ; RSA512_NULL_MD5
02,  ; RSA512_NULL_SHA
08,  ; RSA512_DES40CBC_SHA
09,  ; RSA1024_DESCBC_SHA
0A,  ; RSA1024_3DESCBC_SHA
62,  ; RSA1024_EXPORT_DESCBC_SHA
2F,  ; TLS_RSA_WITH_AES_128_CBC_SHA
35   ; TLS_RSA_WITH_AES_256_CBC_SHA
```

D.5 STATUS Command

Hier ist das STATUS Command des Connector Server abgedruckt. Diesen Output bekommt man mit dem Consolenkommando `msg r1,data=status`.

```
msg r1,data=status
AR 0015 1I40I  READY
R1 0045 IESC1029I STATUS COMMAND
R1 0045     SERVER CONFIG FILE     = DD:PRD2.CONFIG(IESVCSRV.Z)
R1 0045   CONFIGURATION INFORMATION:
R1 0045     MAX NUM. OF CLIENTS    = 256
R1 0045     TCP/IP SERVER PORT     = 2893
R1 0045     SSL ENABLED            = NO
R1 0045     SECURITY               = FULL
R1 0045     ASCII CODEPAGE         = IBM-850
R1 0045     EBCDIC CODEPAGE        = IBM-1047
R1 0045     DESCRIPTION STRING     = VSE CONNECTOR SERVER ON T29LPAR6
R1 0045     TRACING LEVEL          = 0x00000000
R1 0045     TRACE OUTPUT FILE      = DD:SYSLOG
R1 0045     LIBRARIAN CONFIG FILE  = DD:PRD2.CONFIG(IESLIBDF.Z)
R1 0045     USERS CONFIG FILE      = DD:PRD2.CONFIG(IESUSERS.Z)
R1 0045     PLUGIN CONFIG FILE     = DD:PRD2.CONFIG(IESPLGIN.Z)
```

```
R1 0045     SSL CONFIG FILE        = DD:PRD2.CONFIG(IESSSLCF.Z)
R1 0045   GENERAL INFORMATION:
R1 0045     VSE SYSTEM RELEASE     = 4.1.0
R1 0045     VSE SYSTEM DATE        = 26.2.2007
R1 0045   LIST OF CLIENTS:
R1 0045      NO CLIENTS
R1 0045   SUMMARY INFORMATION:
R1 0045     NUM. CLIENTS           = 0
R1 0045     NUM. RUNNING CLIENTS   = 0
R1 0045     NUM. SEND/RECV CLIENTS = 0
R1 0045     NUM. WAITING CLIENTS   = 0
R1 0045     NUM. CLIENTS TO SETUP  = 0
R1 0045     NUM. CLIENTS TO DELETE = 0
R1 0045   PLUGIN INFORMATION:
R1 0045     NUM PLUGINS     = 5
R1 0045     PLUGIN  (ID: 0):
R1 0045      PHASE-NAME     = IESSAPLG
R1 0045      PARM-STRING    = CICS=F2,CONS=IESA,TRANS=IEXM,EXIT=IESSAEXT
R1 0045      DESCRIPTION    = System Activity Plugin. Copyright (C) 1998 by
R1 0045     BM Corp.
R1 0045      VERSION = 1.0
R1 0045      NUM COMMANDS   = 1
R1 0045      COMMAND-ID    = 0x88000010
R1 0045      PLUGIN  (ID: 1):
R1 0045      PHASE-NAME     = IESHTOHP
R1 0045      PHASE-NAME     = IESHTOHP
R1 0045      PARM-STRING    =
R1 0045      DESCRIPTION    = Host to Host Transfer Plugin
R1 0045      VERSION        = 1.0
R1 0045      NUM COMMANDS   = 2
R1 0045       COMMAND-ID    = 0x88000020
R1 0045       COMMAND-ID    = 0x88000021
R1 0045     PLUGIN  (ID: 2):
R1 0045      PHASE-NAME     = IESCOMPH
R1 0045      PARM-STRING    =
R1 0045      DESCRIPTION    = Compile Helper Plugin (C) 1999 by IBM
R1 0045      VERSION        = 1.0
R1 0045      NUM COMMANDS   = 1
R1 0045       COMMAND-ID    = 0x88000003
R1 0045     PLUGIN  (ID: 3):
R1 0045      PHASE-NAME     = IESVSAPL
R1 0045      PARM-STRING    =
R1 0045      DESCRIPTION    = VSAM API Plugin (C) 2000 by IBM
R1 0045      VERSION        = 1.0
R1 0045      NUM COMMANDS   = 4
R1 0045       COMMAND-ID    = 0x88000030
R1 0045       COMMAND-ID    = 0x88000031
R1 0045       COMMAND-ID    = 0x88000032
R1 0045       COMMAND-ID    = 0x88000033
R1 0045     PLUGIN  (ID: 4):
R1 0045      PHASE-NAME     = IESDLIPL
R1 0045      PARM-STRING    =
R1 0045      DESCRIPTION    = Remote DLI Plugin (C) 2000 by IBM
```

```
R1 0045      VERSION        = 2.0
R1 0045      NUM COMMANDS   = 2
R1 0045       COMMAND-ID    = 0x88000034
R1 0045       COMMAND-ID    = 0x88000035
R1 0045   VSAM INFORMATION:
R1 0045     AUTOCLOSE TIME        = 600
R1 0045     NUM OPENED CLUSTERS   = 0
R1 0045     NUM ALLOCATED SLOTS   = 0
```

E TCP/IP Konfigurationsdatei

TCP/IP Konfigurationsdatei **IPINIT00.L**

```
*----------------------------------------*
*                                        *
*        DEFINE THE CONSTANTS            *
*                                        *
*----------------------------------------*
SET IPADDR   = xxx.xxx.xxx.xxx
SET MASK     = 255.255.252.000
*
SET ALL_BOUND        = 30000
SET WINDOW           = 65535
SET TRANSFER_BUFFERS = 20
SET MAX_BUFFERS      =  6
SET TELNETD_BUFFERS  = 20
SET RETRANSMIT       = 100
SET DISPATCH_TIME    = 30
SET REDISPATCH       = 10
SET CONSOLE_HOLD     = ON
*
SET GATEWAY          = ON
*----------------------------------------*
*                                        *
*        WAIT FOR VTAM STARTUP           *
*                                        *
*----------------------------------------*
WAIT     VTAM
*----------------------------------------*
*                                        *
*        DEFINE THE COMMUNICATION LINKS  *
*                                        *
*----------------------------------------*
*
DEFINE LINK,ID=OSAFE,TYPE=OSAX,DEV=518,DATAPATH=51A,MTU=1492,  -
PORTNAME=OSAPORT
SET LINK_RETRY              = 18000
*
*----------------------------------------*
*                                        *
*        DEFINE ROUTINE INFORMATION      *
*                                        *
*----------------------------------------*
DEFINE ROUTE,ID=NETZ,     LINKID=OSAFE,        IPADDR=9.152.24.0
```

```
DEFINE ROUTE,ID=ALL,      LINKID=OSAFE,        IPADDR=0.0.0.0,  -
GATEWAY=9.152.24.1
*-----------------------------------------*
*                                         *
*        DEFINE TELNET DAEMONS            *
*                                         *
*-----------------------------------------*
*DEFINE TELNETD,ID=LU,TERMNAME=TELNLU,TARGET=DBDCCICS,PORT=23,COUNT=3, -
*       LOGMODE=S3270,LOGMODE3=D4B32783,LOGMODE4=D4B32784, -
*       LOGMODE5=D4B32785,POOL=YES
SET CONNECT_SEQUENCE     =OFF
* DEFINE TELNETD,ID=LU21,TERMNAME=TELNLU21,TARGET=DBDCCICS,PORT=23
* DEFINE TELNETD,ID=LU22,TERMNAME=TELNLU22,TARGET=DBDCCICS,PORT=23
*-----------------------------------------*
*                                         *
*        DEFINE FTP DAEMONS               *
*                                         *
*-----------------------------------------*
DEFINE FTPD,ID=FTP,PORT=21,COUNT=2
DEFINE FTPD,ID=FTP11,PORT=21
DEFINE FTPD,ID=FTP12,PORT=21
*-----------------------------------------*
*                                         *
*        DEFINE HTTP DAEMONS              *
*                                         *
*-----------------------------------------*
* DEFINE HTTPD,ID=MANUS,ROOT='PRD2.TEST'
*-----------------------------------------*
*                                         *
*        LINE PRINTER DAEMONS             *
*                                         *
*-----------------------------------------*
* DEFINE LPD,PRINTER=FAST,QUEUE='POWER.LST.A'
* DEFINE LPD,PRINTER=FASTLIB,QUEUE='PRD2.SAVE'
* DEFINE LPD,PRINTER=LOCAL,QUEUE='POWER.LST.A',LIB=PRD2,SUBLIB=SAVE
*-----------------------------------------*
*                                         *
*        AUTOMATED LINE PRINTER CLIENT    *
*                                         *
*-----------------------------------------*
DEFINE EVENT,ID=LST_LISTEN,TYPE=POWER,CLASS=X,QUEUE=LST,ACTION=LPR
*-----------------------------------------*
*                                         *
*        SETUP THE FILE SYSTEM            *
*                                         *
*-----------------------------------------*
DEFINE FILESYS,LOCATION=SYSTEM,TYPE=PERM
*
DEFINE FILE,PUBLIC='IJSYSRS',DLBL=IJSYSRS,TYPE=LIBRARY
DEFINE FILE,PUBLIC='PRD1',DLBL=PRD1,TYPE=LIBRARY
DEFINE FILE,PUBLIC='PRD2',DLBL=PRD2,TYPE=LIBRARY
DEFINE FILE,PUBLIC='POWER',DLBL=IJQFILE,TYPE=POWER
*
MODIFY FILE,PUBLIC='VSE.SYSRES.LIBRARY',TYPE=LIBRARY
```

```
MODIFY FILE,PUBLIC='VSE.PRD1.LIBRARY',TYPE=LIBRARY
MODIFY FILE,PUBLIC='VSE.PRD2.LIBRARY',TYPE=LIBRARY
MODIFY FILE,PUBLIC='VSE.DUMP.LIBRARY',TYPE=LIBRARY
MODIFY FILE,PUBLIC='VSE.PRIMARY.LIBRARY',TYPE=LIBRARY
*
MODIFY FILE,PUBLIC='ICCF.LIBRARY',TYPE=ICCF
MODIFY FILE,PUBLIC='VSE.POWER.QUEUE.FILE',TYPE=POWER
*-----------------------------------------------*
*                                               *
*     MESSAGE SETTINGS                          *
*                                               *
*-----------------------------------------------*
SET MESSAGE CRITICAL         = ON
SET MESSAGE WARNING          = ON
SET MESSAGE VITAL            = ON
SET MESSAGE IMPORTANT        = ON
SET MESSAGE INFORMATION      = ON
SET PING_MESSAGE             = ON
*-----------------------------------------------*
*                                               *
*     PERMITTED USERIDS                         *
*                                               *
*-----------------------------------------------*
DEFINE USER,ID=SYSA,PASSWORD=TEEM01,DATA='VSE/ESA SYSA'
DEFINE USER,ID=SYSB,PASSWORD=TEEM01,DATA='VSE/ESA SYSB'
DEFINE USER,ID=SYSC,PASSWORD=TEEM01,DATA='VSE/ESA SYSC'
DEFINE USER,ID=OPER,PASSWORD=TEEM01,DATA='VSE/ESA OPER'
*-----------------------------------------------*
*                                               *
*     SECURITY SETTINGS                         *
*                                               *
*-----------------------------------------------*
DEFINE SECURITY,DRIVER=BSSTISX
SET SECURITY                 = ON
SET SECURITY_ARP             = OFF
SET SECURITY_IP              = OFF
SET ISOLATION                = OFF
*-----------------------------------------------*
*                                               *
*       OPERATION AND STARTUP                   *
*                                               *
*-----------------------------------------------*
SET CONSOLE_HOLD            = OFF
SET RECORD                  = OFF
*-----------------------------------------------*
*                                               *
*       SYMBOLIC NAMES                          *
*                                               *
*-----------------------------------------------*
*-----------------------------------------------*
*                                               *
*       START HTTP SERVER                       *
*                                               *
*-----------------------------------------------*
```

```
* DEFINE HTTPD,ID=MYHTTPD,ROOT=PRIMARY.SYSA
*-----------------------------------------*
*                                         *
*        SETUP MEMBER NETWORK.L TO        *
*        EXECUTE ONCE THE ENGINE HAS      *
*        BEEN ACTIVATED                   *
*                                         *
*-----------------------------------------*
INCLUDE NETWORK,DELAY
```

E.1 Q SET Command

Dieser Output wird von dem Kommando `104 q set` erzeugt und gibt alle TCP/IP Einstellungen und Konfigurationen aus.

```
F7-0104 IPN300I Enter TCP/IP Command
104 q set
F7 0098 IPN253I << TCP/IP Current Options >>
F7 0098 IPN452I System ID: 00
F7 0098 IPN453I IP Address: 9.152.84.147 Submask: 255.255.252.0
F7 0098 IPN454I All Bound Time: 30000 Link Retry Time: 18000
F7 0098 IPN456I Retransmit Time: 100 Window: 65535
F7 0098 IPN518I Receive MSS: 32684
F7 0098 IPN458I Console Hold: Off Record: Off
F7 0098 IPN459I Gateway: On Isolation: Off
F7 0098 IPN804I Telnet translation will use System Default
F7 0098 IPN806I Checksum: Software Connect_Sequence: Off
F7 0098 IPN806I Downcheck: On Dynamic_Route: On
F7 0098 IPN806I Fixed_Retransmit: Off Full_CETI: On
F7 0098 IPN806I Full_Critical: Off Local_DLBL: On
F7 0098 IPN806I Match_Message: On Ping_Message: On
F7 0098 IPN806I Reuse: On Separator_Pages: Off
F7 0098 IPN806I SDOpen_Extra: Off Spincheck: On
F7 0098 IPN806I Sockcheck: On
F7 0098 IPN806I Traffic: On Singledest: On
F7 0098 IPN806I ARPdelete: Off ARP_Time: 90000
F7 0098 IPN806I Auto_time: 9000 Pulse_Time: 18000
F7 0098 IPN806I Console_Port: 0
F7 0098 IPN806I Additional_Window: 100 Reuse_Size: 10
F7 0098 IPN806I Window_Restart: 10 Window_Depth: 30
F7 0098 IPN806I Slow_Start: 10 Slow_Restart: 10
F7 0098 IPN806I Slow_Increment: 1
F7 0098 IPN806I Default_Domain:
F7 0098 IPN806I Close Connection Depth: 10 Separator Page Count: 0
F7 0098 IPN806I FTPBATCH_Fetch: off
F7 0098 IPN806I Subtask_OPEN: off
F7 0098 IPN806I Memory Verification: Off Connection Queuing Off
F7 0098 IPN806I Buffer Validation: Off POWERUSERID: SYSTCPIP
F7 0098 IPN806I ListIDCAMS: On AutoLoad: Off
F7-0104 IPN300I Enter TCP/IP Command
```

F Jobs und Example Output

F.1 Benutzerprofil

Benutzerverwaltung: DTSECTAB Zugriffskontrollklassen und Gruppenzugehörigkeit

```
 IESADMUPR1           ADD OR CHANGE RESOURCE ACCESS RIGHTS
Base     II       CICS     ResClass ICCF

     Place an 'X' next to the transaction security keys for user ENDU
   01 X   02 X   03 X   04 X   05 X   06 X   07 X   08 X   09 X   10 X   11 X
   12 X   13 X   14 X   15 X   16 X   17 X   18 X   19 X   20 X   21 X   22 X
   23 X   24 X   25 X   26 X   27 X   28 X   29 X   30 X   31 X   32 X   33 X
   34 X   35 X   36 X   37 X   38 X   39 X   40 X   41 X   42 X   43 X   44 X
   45 X   46 X   47 X   48 X   49 X   50 X   51 X   52 X   53 X   54 X   55 X
   56 X   57 X   58 X   59 X   60 X   61 X   62 X   63 X   64 X

     Specify the access rights for 1-32 DTSECTAB access control classes
     ( _-No access, 1-Connect, 2-Read, 3-Update, 4-Alter )
   01 _   02 _   03 _   04 _   05 _   06 _   07 _   08 _   09 _   10 _   11 _
   12 _   13 _   14 _   15 _   16 _   17 _   18 _   19 _   20 _   21 _   22 _
   23 _   24 _   25 _   26 _   27 _   28 _   29 _   30 _   31 _   32 _

   READ DIRECTORY..... 1   User can read directory with Connect (1-yes, 2-no)
   B-TRANSIENTS....... 1   User can manipulate B-Transients     (1-yes, 2-no)

PF1-HELP                    3-END                     5-UPDATE
PF7-BACKWARD   8-FORWARD
```

Abbildung F.1: siehe [zVSEAd07] - Zugriffskontrollklassen für DTSECTAB Einträge.

F.2 BSTADMIN JOBS

Hier werden die BSTADMIN Jobs, die zur Testkonfiguration verwendet wurden gezeigt.

F.2.1 Defines Profiles

```
* $$ JOB JNM=BSTADMIN,CLASS=O,DISP=D
```

```
// JOB BSTADMIN
// PAUSE ID STMT OR ENTER
*
* * This is a sample job to execute BSTADMIN commands
* * It defines profiles to test AUDIT on access level
*
// EXEC BSTADMIN
* User HUGO should be used for tests with READ access
AD FACILITY MY.TEST01F AUDIT(NONE) DA('AUDIT Test ')
AD FACILITY MY.TEST01S AUDIT(NONE) UACC(READ) DA('AUDIT Test ')
AD FACILITY MY.TEST02F AUDIT(ALL) DA('AUDIT Test ')
AD FACILITY MY.TEST02S AUDIT(ALL) UACC(READ) DA('AUDIT Test ')
AD FACILITY MY.TEST03F AUDIT(SUCC) DA('AUDIT Test ')
AD FACILITY MY.TEST03S AUDIT(NONE) UACC(READ) DA('AUDIT Test ')
AD FACILITY MY.TEST04F AUDIT(FAIL) DA('AUDIT Test ')
AD FACILITY MY.TEST04S AUDIT(ALL) DA('AUDIT Test ')
AD FACILITY MY.TEST05F AUDIT(SUCC) DA('AUDIT Test ')
AD FACILITY MY.TEST05S AUDIT(ALL) UACC(READ) -
DA('AUDIT Test ')
AD FACILITY MY.TEST06F AUDIT(SUCC) DA('AUDIT Test ')
AD FACILITY MY.TEST06S AUDIT(SUCC) UACC(READ) -
DA('AUDIT Test ')
* User HUGO should be used for tests with UPD access
AD FACILITY MY.TEST07F AUDIT(SUCC) DA('AUDIT Test ')
PE FACILITY MY.TEST07F ID(HUGO) ACC(READ)
AD FACILITY MY.TEST07S AUDIT(SUCC) UACC(READ) -
DA('AUDIT Test ')
PE FACILITY MY.TEST07S ID(HUGO) ACC(UPD)
AD FACILITY MY.TEST08F AUDIT(SUCC) DA('AUDIT Test ')
PE FACILITY MY.TEST08F ID(HUGO) ACC(READ)
AD FACILITY MY.TEST08S AUDIT(SUCC) UACC(READ) -
DA('AUDIT Test ')
PE FACILITY MY.TEST08S ID(HUGO) ACC(UPD)
*
* After all transactions are defined, you should activate them
PF D R
/*
/&
* $$ EOJ
```

F.2.2 Define Users

```
* $$ JOB JNM=UPDATECF,CLASS=0,DISP=D
* $$ PUN DISP=I
// JOB UPDATECF
// EXEC PROC=DTRICCF
// DLBL IESCNTL,'VSE.CONTROL.FILE',VSAM,CAT=VSESPUC
// EXEC IESUPDCF,SIZE=64K
ICCF=IGNORE
ADD ASYS,EX10TION,SYSA,REVOKE=0
ALT ASYS,DAYS=180
ADD ASYSTES1,EX10TION,SYSA,REVOKE=0
```

```
ALT ASYSTES1,DAYS=180
ADD APROTES1,EX10TION,PROG,REVOKE=0
ALT APROTES1,DAYS=180
/*
/&
* $$ EOJ
```

F.2.3 Define DTSECTAB

```
* $$ JOB JNM=RRTST1,CLASS=0,DISP=D
// JOB RRTST1
// ID USER=FORSEC,PWD=FORSEC
*
* * The following is a sample job with BSTADMIN
* * to add discrete profiles
*
// DLBL BSTCNTL,'VSE.BSTCNTL.FILE',VSAM,CAT=VSESPUC
// EXEC BSTADMIN
AD ACICSPCT ABC1 UACC(NONE)
AD ACICSPCT 'PREF.ABC1' UACC(NONE)
AD ACICSPCT 'PREF.ABC' UACC(NONE) GEN
PE ACICSPCT 'PREF.ABC' GEN ID(HUGO) A(READ)
AD ACICSPCT ABC UACC(NONE) GEN
PE ACICSPCT ABC ID(MYGRP2) GEN A(UP)
AD ACICSPCT A000 UACC(NONE)
AD ACICSPCT A001 UACC(NONE)
AD ACICSPCT A002 UACC(NONE)
/*
/&
* $$ EOJ
```